**超市库存管理系统需求规格说明书**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**目录**

目录

[1 引言 4](#_Toc59443964)

[1.1项目背景 4](#_Toc59443965)

[1.2 术语定义 4](#_Toc59443966)

[2 应用环境 5](#_Toc59443967)

[2.1 硬件环境 5](#_Toc59443968)

[2.2 软件环境 5](#_Toc59443969)

[2.3 应用环境 5](#_Toc59443970)

[3 基于UML分析系统功能需求 5](#_Toc59443971)

[3.1 用例图 5](#_Toc59443972)

[3.2活动图 6](#_Toc59443973)

[3.3类图 9](#_Toc59443974)

[3.4 顺序图 9](#_Toc59443975)

[3.5 通信图 11](#_Toc59443976)

[3.6状态图 12](#_Toc59443977)

[4 性能需求 13](#_Toc59443978)

[4．1精度 13](#_Toc59443979)

[4.2时间特性要求 13](#_Toc59443980)

[4.3灵活性 14](#_Toc59443981)

[5 系统实现 14](#_Toc59443982)

[5.1 管理员进入系统 14](#_Toc59443983)

[5.2注册新用户功能 16](#_Toc59443984)

[5.3获取管理员用户界面截图 19](#_Toc59443985)

[5.4用户信息模糊查询界面截图 20](#_Toc59443986)

[5.5商品入库界面截图 21](#_Toc59443987)

[5.6新建商品信息界面截图 22](#_Toc59443988)

[5.7获取供应商界面截图 24](#_Toc59443989)

[5.8删除界面截图 25](#_Toc59443990)

[5.9出库界面截图 27](#_Toc59443991)

[5.10出入库报表管理界面截图 28](#_Toc59443992)

[5.11商品管理界面截图 31](#_Toc59443993)

[5.12警告界面截图 32](#_Toc59443994)

[6 总结 37](#_Toc59443995)

# 1 引言

校园超市的库存物资管理往往是很复杂、很繁琐的。由于所掌握的物资种类众多,订货、管理的渠道各有差异,各个校园超市之间的管理体制不尽相同,因此仓库的库存管理必须编制一套库存信息管理系统,用计算机对其管理,同时要依据校园超市自身的实际处境来制作对应的处理方法。超市库存管理系统很好的解决了上述问题,它依靠现代化的计算机信息处理技术来管理超市,从而节省了大量的人力、物力,减轻了劳动强度,并且能够快速反映出商品入库、出库、存货等状况,使管理人员快速对市场的变化做出相应的决策,提高超市经营管理效率。

## 1.1项目背景

本宝鸡文理学院超市库存管理系统软件需求分析报告是为学校超市系统而编写的。一套适用于宝鸡文理学院的超市进行较多种类货物的库存管理系统,力争以最简洁的方式帮助超市管理人员进行入库、出库操作和库存管理,提供表格导出等功能。本宝鸡文理学院超市库存管理报告旨在使超市商品管理人员明确超市库存管理系统的要求和细节，给开发者（18级软工二班代登辉、韩思琦、陈培陪、李彤彤及18级软工一班杨玉芳）了解需求实现的难度和困难，最终提供给老师（审核人）和超市商品管理人员（管理者）讨论和审核，达到沟通效果。

对于现今的大学生超市仓库管理非常繁琐,管理人员需付出大量的工作时间来统计商品的数量和保证到期日期早的商品能及时上架,工作效率很低。因此为提高工作效率,减轻货物管理人员的工作负担,决定开发商品仓库管理系统软件。该商品仓库管理系统的开发主要是为了解决超市商品库存的缺货问题,保证超市商品供销链上的及时性与动态性。

## 1.2 术语定义

**专门术语:**

静态数据--系统固化在内的描述系统实现功能的一部分数据。

动态数据--在软件运行过程中用户输入的后系统输出给用户的一部分数据,也就是系统要处理的数据。

数据字典--数据字典中的名字都是一些属性与内容的抽象和概括,它们的特点是数据的“严密性”和“精确性”。

# 2 应用环境

## 2.1 硬件环境

列出运行该软件所需要的硬设备。说明其中的新型设备及其专门功能，包括：

1. CPU主频1.6GHZ以上及内存1G以上；
2. 硬盘150G以上；
3. 操作系统：windows10
4. 数据库服务器：Microsoft SQL Server2008

## 2.2 软件环境

操作系统windows系统，数据库管理系统：SQL数据库系统，QTCreator编译工具。

## 2.3 应用环境

Windows 10系统

# 3 基于UML分析系统功能需求

## 3.1 用例图

用例图从用户角度描述系统功能。该用例图描述系统的参与者仓库管理员与系统的登录、用户管理、供货商管理、商品管理、入库管理、出库管理、报表管理等用例之间的关系。

系统用例图入图1所示：



**图 1 用例图**

## 3.2活动图

活动图是描述活动的顺序。此系统的活动图分为：出库活动图和入库活动图，它们各自描述了各自与系统进行交互的流程操作。

该出库活动图描述了仓库管理员与系统进行交互的流程操作。

出库活动图如图2所示：



**图 2出库活动图**

该入库活动图描述了仓库管理员与系统进行交互的流程操作。

入库活动图如图3所示：



**图3 入库活动图**

## 3.3类图

类图是描述类，接口以及它们之间关系的图，是一种静态模型，显示了系统中各个类的静态结构。该类图描述超市库存管理系统中的相关类以及各个类的关系来描述静态视图，为的是让开发人员能够更加清楚了解组成该系统的结构。

超市库存管理系统类图如图4所示：



**图 4 超市库存管理系统类图**

## 3.4 顺序图

顺序图代表了一个相互作用，在以时间为次序的对象之间的通信集合。

**出库顺序图：**该顺序图描绘了仓库管理员、出库、数据库之间传递消息的时间顺序，以此来表示出库的行为顺序。

出库顺序图如图5所示：

**图 5出库顺序图**

**入库顺序图：**该顺序图描绘了仓库管理员、入库、数据库之间传递消息的时间顺序，以此来表示入库的行为顺序。

入库顺序图如图6所示：



**图6 入库顺序图**

## 3.5 通信图

出库通信图如图7所示：



**图7出库通信图**

出库通信图如图8所示：



**图8入库通信图**

## 3.6状态图

状态图主要描述了系统在各个状态之间的转换关系。在进行商品入库操作时,通过点击界面上的各个菜单跳转到各个不同的页面,显示商品的不同状态，各个操作执行完成之后都可以直接返回到系统操作界面,执行其他操作。超市库存管理系统状态图如图9所示:



**图 9 状态图**

# 4 性能需求

## 4．1精度

输入的数据必须符合设计的格式要求：

1. 产品数量必须为正整数；
2. 商品金额属性，小数点后面取前两位。

## 4.2时间特性要求

说明对于该软件的时间特性要求：

1. 响应时间0.1s；
2. 更新处理时间0.1s；
3. 数据的转换和传送时间0.1s；
4. 解题时间；等待的要求0.5s。

要求登陆系统时,系统的响应时间控制在一秒内，更新的时间越低越好, 数据传送时间尽量缩短。

## 4.3灵活性

该系统是以C++语言为背景的，应该是可以跨平台运行的，至于操作系统应该也是可以转换在windows操作系统。数据库是SQL Server。

如果需求发生某些变化时，该软件对这些变化的适应能力如下：

1. 操作方面添加部分表单；
2. 运行环境不变；
3. 同其他软件接口不变；
4. 精度和有效时限不变；
5. 计划的变化或改进。

# 5 系统实现

## 5.1 管理员进入系统

系统里已写入默认管理员，输入用户名、密码，即可进入该系统。管理员进入系统界面截图如图10所示。



**图10 登录界面截图**

bool Mysql::**verifyLogin**(const QString name,const QString password)

{

QSqlQuery query;

query.exec("select \* from [user] where exist = 1");

bool chekname=false;

bool chekpassword=false;

// 打印输出用户名和密码

while (query.next()) {

qDebug() << query.value(0).toInt()

<< query.value(1).toInt()

<< query.value(2).toString();

if(query.value(1).toInt() ==1 ){

chekname = (QString::compare(name,query.value(2).toString(),

Qt::CaseSensitive) == 0);

chekpassword = (QString::compare(password,query.value(3).toString(),

Qt::CaseSensitive) == 0);

}else{

continue;

}

if(chekname&&chekpassword){

return true;

}

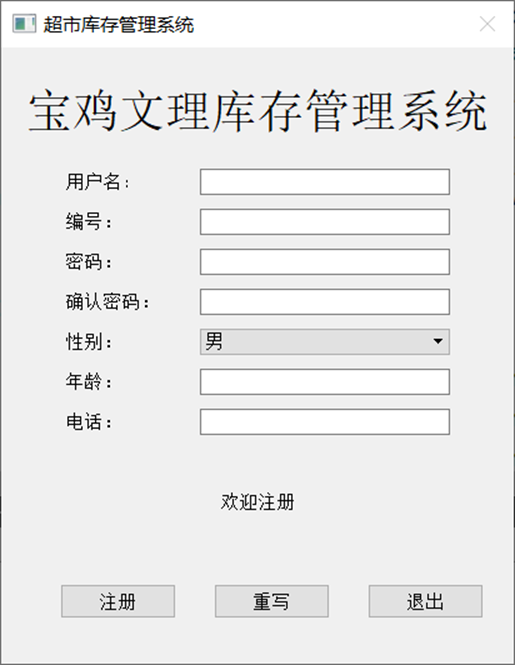
}

return false;

}

## 5.2注册新用户功能

管理员进入系统后可添加新的用户或者注册新的管理员。注册界面截图如图11所示：



**图11注册界面截图**

// 当注册页面的注册按下

void Register::on\_pushButton\_login\_clicked()

{

Mysql sql;

if(sql.isUid(ui->lineEdit\_id->text().toInt())){

ui->label\_tip->clear();

ui->label\_tip->setStyleSheet(QString("color:red;"));

ui->label\_tip->setText(QString("编号已存在"));

return;

}

// 用户名是否存在

if(sql.isUname(ui->lineEdit\_uname->text())!=-1){

ui->label\_tip->clear();

ui->label\_tip->setStyleSheet(QString("color:red;"));

ui->label\_tip->setText(QString("用户名已存在"));

return;

}

// 两次密码是否一致

if((QString::compare(ui->lineEdit\_password->text(),ui->lineEdit\_password\_2->text(),

Qt::CaseSensitive) == 0))

{

User user;

user.setId(ui->lineEdit\_id->text().toInt());

user.setUname(ui->lineEdit\_uname->text());

user.setUpassword(ui->lineEdit\_password->text());

user.setExit(1);

int age = ui->lineEdit\_age->text().toInt();

if(age>0&&age<200){ // 年龄

user.setAge(age);

}else{

ui->label\_tip->clear();

ui->label\_tip->setStyleSheet(QString("color:red;"));

ui->label\_tip->setText(QString("年龄错误"));

return;

}

user.setSex(ui->combo\_sex->currentText());

user.setTel(ui->lineEdit\_tel->text());

if(sql.applyAccount(user)){

user.printUser();

QMessageBox::warning(nullptr, "注意", "用户创建成功");

on\_pushButton\_clear\_clicked();

}

}else { // 两次密码不一样

ui->label\_tip->clear();

ui->label\_tip->setStyleSheet(QString("color:red;"));

ui->label\_tip->setText(QString("两次密码不一致"));

}

}

## 5.3获取管理员用户界面截图

仓库管理员可以获取用户管理员界面如图12所示：



**图12获取用户界面图**

std::vector<User>\* Mysql::getUsers()

{

std::vector<User> \*List = new std::vector<User>;

QSqlQuery query;

query.exec("select \* from [user] where exist = 1");

while (query.next()) {

User user;

user.setId(query.value(0).toInt());

user.setExit(query.value(1).toInt());

user.setUname(query.value(2).toString());

user.setUpassword(query.value(3).toString());

user.setSex(query.value(4).toString());

user.setAge(query.value(5).toInt());

user.setTel(query.value(6).toString());

user.setTime(query.value(7).toDateTime());

List->push\_back(user);

}

return List;

}

## 5.4用户信息模糊查询界面截图

默认管理员可以通过模糊信息查询用户信息，用户信息模糊查询界面如图13所示：



**图13模糊查询界面图**

## 5.5商品入库界面截图

管理员进入商品入库界面，选择供应商、输入商品名称等一系列信息，并选择过期时间。商品入库界面如图14所示：



**图14商品入库界面图**

void InForm::**on\_pushButton\_in\_clicked**()

{

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

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

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

int id = sql.getGoodCid(sname,catename,cname);

if(id == 0){

// 创建商品

ui->label\_tip->clear();

ui->label\_tip->setStyleSheet(QString("color:red;"));

ui->label\_tip->setText(QString("请先创建新商品"));

return ;

}

int num = ui->spinBox\_num->text().toInt();

float price = ui->doubleSpinBox\_price->text().toFloat();

float weight = ui->doubleSpinBox\_\_kg->text().toFloat();

if(num <= 0 || price <= 0 || weight <= 0){

ui->label\_tip->clear();

ui->label\_tip->setStyleSheet(QString("color:red;"));

ui->label\_tip->setText(QString("商品数量、价格、质量不能为零"));

return ;

}

QString selftime = ui->dateTimeEdit->dateTime().toString("yyyy-MM-dd hh:mm:ss");

QString tmpSql = QString("INSERT INTO [dbo].[in] ([cid], [price], [time], "

"[shelftime], [num], [uid], [weight] )"

"VALUES"

"( %1, %2, GETDATE(), '%3', %4, %5, %6 )")

.arg(id).arg(price).arg(selftime).arg(num).arg(uid).arg(weight);

qDebug() << tmpSql;

sql.Myexec(tmpSql);

QMessageBox::warning(nullptr, "提示", "入库成功");

ui->label\_tip->clear();

}

## 5.6新建商品信息界面截图

管理员进入新建商品界面，依次添加信息、选择商品类别、供应商即可成功添加新的商品。新建商品信息界面如图15所示：



**图15新建商品信息界面图**

void CommodityAddForm::**on\_pushButton\_new\_clicked**()

{

QString name = ui->lineEdit\_cname->text();

QString category = ui->comboBox\_category->currentText();

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

if(suppliername.isEmpty()){

this->ui->label\_tip->setText("请先选择供应商");

return;

}

int sid = sql.SupplierID(suppliername);

int maxnum = ui->lineEdit\_MAX->text().toInt();

int minnum = ui->lineEdit\_MIN->text().toInt();

QString tmp = QString("INSERT INTO [dbo].[commodity]([cname],"

" [sid], [category], [minnum], [maxmun]) "

"VALUES ('%1', %2, '%3', %4, %5)")

.arg(name).arg(sid).arg(category)

.arg(minnum).arg(maxnum);

qDebug() << tmp;

sql.Myexec(tmp);

suppliercomboBox();

QMessageBox::warning(nullptr, "提示", "商品信息创建成功");

}

## 5.7获取供应商界面截图

进入供应商界面即可获取所有的供应商信息，输入供应商任一信息即可相对应的供应商信息。获取供应商界面如图16所示：

**图16获取供应商界面图**

std::vector<Supplier> \*Mysql::**getSupplier**(QString name)

{

std::vector<Supplier> \*list = new std::vector<Supplier>;

QSqlQuery query;

query.exec(QString("select \* from [supplier] where exist = 1 and sname LIKE '%%1%'").arg(name));

while (query.next()) {

Supplier supplier;

supplier.setSid(query.value(0).toInt());

supplier.setExist(query.value(1).toInt());

supplier.setSname(query.value(2).toString());

supplier.setAddr(query.value(3).toString());

supplier.setTel(query.value(4).toString());

list->push\_back(supplier);

}

return list;

}

## 5.8删除界面截图

点击删除供应商的任一信息即可删除该供应商对应的所有信息。删除供应商界面如图17所示：



**图17删除供应商界面图**

void SupplierForm::**on\_pushButton\_delete\_clicked**()

{

int row = ui->tableView->currentIndex().row();

int id = this->model->*data*(this->model->*index*(row,0)).toInt();

list[size] =id;

qDebug() << list[size] << "row = " << row;

size++;

model->removeRow(ui->tableView->currentIndex().row());

}

// 点击修改供应商

void SupplierForm::**on\_pushButton\_alter\_clicked**()

{

int row = ui->tableView->currentIndex().row();

Supplier supplier;

getSupplier(row,supplier);

supplier.printSupplier();

upadteSupplier(supplier);

}

## 5.9出库界面截图

管理员进入商品出库界面，选择供应商以及商品类别即可。出库界面如图18所示：



**图18出库界面图**

void OutForm::**on\_pushButton\_out\_clicked**()

{

int row = ui->tableView->currentIndex().row();

int id = this->model->*data*(this->model->*index*(row,0)).toInt();

int num = this->model->*data*(this->model->*index*(row,2)).toInt();

int outnum = this->model->*data*(this->model->*index*(row,8)).toInt();

if(num < outnum){

QMessageBox::warning(nullptr, "警告", "出库数量超过库存");

this->model->setItem(row, 8,new QStandardItem(QString("")));

return;

}

if(outnum <= 0){

QMessageBox::warning(nullptr, "警告", "请输入出库数量");

return;

}

int out = num - outnum;

QString tmp = QString("UPDATE [dbo].[in] "

"SET [num] = %1 WHERE [iid] = %2")

.arg(out).arg(id);

sql.outgood(id,outnum,uid);

qDebug() << tmp;

sql.Myexec(tmp);

QMessageBox::warning(nullptr, "提示", "出库成功");

on\_pushButton\_refresh\_clicked();

}

## 5.10出入库报表管理界面截图

进出入库界面，在报表类型下拉框中选择入库，在时间下拉框中选择时间段即可导出相应的信息。入库报表管理界面如图19所示：



**图19入库报表管理界面图**

进出入库界面，在报表类型下拉框中选择出库，在时间下拉框中选择时间段即可导出相应的信息。出库报表管理界面如图20所示：



**图20出库报表管理界面图**

void StatementForm::on\_pushButton\_refresh\_clicked()

{

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

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

// 获取当前时间

QDateTime data = QDateTime::currentDateTime();

if(QString::compare(time,QString("一天以内")) == 0){

//data = data.addDays(-1);

time = data.toString("yyyy-MM-dd");

}else if(QString::compare(time,QString("一个月内"))== 0){

data = data.addMonths(-1);

time = data.toString("yyyy-MM-dd");

}else if(QString::compare(time,QString("一个季度内"))== 0){

data = data.addMonths(-3);

time = data.toString("yyyy-MM-dd");

}else if(QString::compare(time,QString("一年以内"))== 0){

data = data.addYears(-1);

time = data.toString("yyyy-MM-dd");

}else{

time = QString("");

}

std::vector<Goods> \*Slist;

if(QString::compare(IO,QString("入库")) == 0){

initable(1);

Slist = sql.getinGoods(time);

int i = 0;

for(std::vector<Goods>::iterator it=(\*Slist).begin();it!=(\*Slist).end();it++){

/\*setItem设置条目栏中的一个格子的信息\*/

setINTable(i,it);

i++;

}

if(Slist!=NULL)

delete Slist;

Slist = NULL;

}else{

initable(2);

Slist = sql.getoutGoods(time);

int i = 0;

for(std::vector<Goods>::iterator it=(\*Slist).begin();it!=(\*Slist).end();it++){

/\*setItem设置条目栏中的一个格子的信息\*/

setOUTable(i,it);

i++;

}

if(Slist!=NULL)

delete Slist;

Slist = NULL;

}

qDebug() << uid << "id id";

qDebug() << data.toString("yyyy-MM-dd") << " " << IO ;

}

## 5.11商品管理界面截图

进入商品信息管理界面，在供应商下拉框中选择供应商，在商品类别下拉框中选择商品类别即可导出相应的信息。商品管理界面如图21所示：



**图21商品管理界面图**

void GoodManangeForm::on\_pushButton\_find\_clicked()

{

initTable();

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

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

qDebug() << sname << " " << catename;

std::vector<Goods> \*Slist = sql.getGoods(sname,catename);

int i = 0;

for(std::vector<Goods>::iterator it=(\*Slist).begin();it!=(\*Slist).end();it++){

/\*setItem设置条目栏中的一个格子的信息\*/

setTable(i,it);

i++;

}

if(Slist!=NULL)

delete Slist;

Slist = NULL;

}

## 5.12警告界面截图

实时更新警告，一分钟更新一次。

警告商品过期界面如图22所示：



**图22警告商品过期界面图**

警告商品即将过期界面如图23所示：



**图23警告商品即将过期界面图**

警告商品库存不足界面如图24所示：



**图24警告商品库存不足界面图**

警告商品库存过量界面如图25所示：



**图25警告商品库存过量界面图**

void WarningForm::**settableGoods**(int i, std::vector<Goods>::iterator s)

{

s->printGoods();

this->model->setItem(i, 0, new QStandardItem(QString("%1").arg(s->getCid())));

this->model->setItem(i, 1, new QStandardItem(QString(s->getCname())));

this->model->setItem(i, 2, new QStandardItem(QString("%1").arg(s->getNum())));

this->model->setItem(i, 3, new QStandardItem(QString(s->getCategory())));

this->model->setItem(i, 4, new QStandardItem(QString(s->getSnmae())));

this->model->setItem(i, 5, new QStandardItem(QString(s->getAddr())));

this->model->setItem(i, 6, new QStandardItem(QString(s->getTel())));

}

void WarningForm::**Overdue**()

{

ui->label->clear();

ui->label->setText(QString("警告商品过期"));

std::vector<Goods> goodlist = getgoods();

int i = 0;

for(std::vector<Goods>::iterator it=goodlist.begin();it!=goodlist.end();it++){

/\*setItem设置条目栏中的一个格子的信息\*/

settable(i,it);

i++;

}

}

void WarningForm::**Expiring**()

{

ui->label->clear();

ui->label->setText(QString("警告商品即将过期"));

std::vector<Goods> goodlist = getExpiringgoods();

int i = 0;

for(std::vector<Goods>::iterator it=goodlist.begin();it!=goodlist.end();it++){

/\*setItem设置条目栏中的一个格子的信息\*/

settable(i,it);

i++;

}

}

void WarningForm::**lack**()

{

ui->label->clear();

ui->label->setText(QString("警告商品库存不足"));

std::vector<Goods> goodlist = getlackgoods();

int i = 0;

for(std::vector<Goods>::iterator it=goodlist.begin();it!=goodlist.end();it++){

/\*setItem设置条目栏中的一个格子的信息\*/

settableGoods(i,it);

i++;

}

}

void WarningForm::**nimiety**()

{

ui->label->clear();

ui->label->setText(QString("警告商品库存过量"));

std::vector<Goods> goodlist = getnimietygoods();

int i = 0;

for(std::vector<Goods>::iterator it=goodlist.begin();it!=goodlist.end();it++){

/\*setItem设置条目栏中的一个格子的信息\*/

settableGoods(i,it);

i++;

}

}

bool WarningForm::**ifOverdue**()

{

QSqlQuery query;

query.exec(QString("SELECT \* from goodsin where shelftime < GETDATE()"));

query.next();

return !query.value(1).toString().isEmpty();

}

bool WarningForm::**ifExpiring**()

{

QSqlQuery query;

query.exec(QString("SELECT \* from goodsin where shelftime < GETDATE()+30 and shelftime > GETDATE()"));

query.next();

return !query.value(1).toString().isEmpty();

}

bool WarningForm::**iflack**()

{

QSqlQuery query;

query.exec(QString("SELECT \* from goodsin where num < minnum"));

query.next();

return !query.value(1).toString().isEmpty();

}

bool WarningForm::**ifnimiety**()

{

QSqlQuery query;

query.exec(QString("SELECT \* from goodsin where num > maxmun"));

query.next();

return !query.value(1).toString().isEmpty();

}