**SRM MEDICAL STORE MANAGEMENT**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**G Gowtham Kumar - AP21110011490**

**Y Tony Babu- AP21110011488**

**B Durga Vinay- AP21110011451**

**Devesh Choudary- AP21110011519**



Under the Guidance of

**Mr. E Bhaskar Santhosh**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[12/2022]**

# Certificate

Date: 21 December 2022

This is to certify that the work present in this Project entitled “**SRM Medical Store Management**” has been carried out by**G Gowtham Kumar, Y Tony Babu, B Durga Vinay and Devesh Chowdary** under Mr. E Bhaskar Santhosh supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

# Table of Contents

[Certificate 1](#_Toc105799088)

[Table of Contents 2](#_Toc105799090)

[Abstract 3](#_Toc105799091)

[1. Introduction 1](#_Toc105799099)

[2. **Objective** 2](#_Toc105799106)

[4. **Implementation** 4](#_Toc105799107)

[5. **Output** 5](#_Toc105799107)

[6. **Concluding Remarks** 7](#_Toc105799107)

# Abstract

The SRM Medical store system software is a medicine stock inventory system used in SRM medical stores that runs for the students and the faculties. This software stores the details of medicine purchase stock and the sell stock. The system should maintain purchase stock detail by company wise and sell stock detail by customer wise. Here the customer is the SRM students and faculties. This software should generate an automated bill for every selling and generate reports for stock, sell, and customer detail.

# Introduction

## This project is used mainly for medical stores to maintain the details of Medical store such as stock and account.

* This medical store management software is so designed as to ease the work load of medical store professionals. The main feature includes inventory and stock control, accounting and client management.
* Project characteristics:

1. Customers management
2. Transition management
3. Sales management
4. Reporting

# Objective

* As this is generic software it can be used by a wide variety of outlets (Retailers and Wholesalers) to automate the process of manually maintaining the record related to the subject of maintaining the stock and cash flows.
* This project is basically updating the manual chemist inventory System To Automated inventory system, So that organization can manage their record in efficient and organized form.
* This software helps you to track all the products of medical shop moreover it's a medical shop accounting software. Flexiable and adaptive software suited to medical shops to medical shops or stores or pharmacies of any size.

# 

# 

**4. Implementation**

**a. Header files:**

**1.addmed.h:**

#ifndef ADDMED\_H

#define ADDMED\_H

#include <QDialog>

namespace Ui {

class Addmed;

}

class Addmed : public QDialog

{

Q\_OBJECT

public:

explicit Addmed(QWidget \*parent = nullptr);

~Addmed();

private slots:

void on\_pushButton\_clicked();

private:

Ui::Addmed \*ui;

};

#endif // ADDMED\_H

**2.buymeddialog.h:**

#ifndef BUYMEDDIALOG\_H

#define BUYMEDDIALOG\_H

#include "showinvoice.h"

#include <QDialog>

#include <QtCore>

#include <QtXml>

#include <QDebug>

#include <iostream>

#include <vector>

using namespace std;

QT\_BEGIN\_NAMESPACE

namespace Ui { class BuyMedDialog; }

QT\_END\_NAMESPACE

class BuyMedDialog : public QDialog

{

Q\_OBJECT

public:

BuyMedDialog(QString Cashier,QWidget \*parent = nullptr);

ShowInvoice \*myShowInvoice;

QString Cashier;

~BuyMedDialog();

private slots:

void on\_generatebill\_button\_clicked();

void on\_addbutton\_clicked();

void on\_search\_button\_clicked();

void on\_delete\_button\_clicked();

void on\_cancel\_button\_clicked();

void WriteTransactionXML();

void EditMedicineList();

private:

Ui::BuyMedDialog \*ui;

};

#endif // BUYMEDDIALOG\_H

**3.CheckLogin.h:**

#ifndef CHECKLOGIN\_H

#define CHECKLOGIN\_H

#include <QtCore>

#include <QtXml>

#include <QDebug>

bool ListElements(QDomElement tagname,QString u\_tocheck,QString p\_tocheck)

{

QDomNodeList items = tagname.elementsByTagName("person");

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

{

QDomNode itemnode = items.at(i);

if(itemnode.isElement())

{

QDomElement itemele=itemnode.toElement();

if(u\_tocheck==itemele.attribute("username"))

{

if(p\_tocheck==itemele.attribute("password"))

{

return 1;

}

}

}

}

return 0;

}

bool CheckLogin(QString role\_text, QString uname, QString pword)

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\loginXML.xml");

//add exception handling

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return -1;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

return -1;

file.close();

}

QDomElement login = document.firstChildElement();

//has to change this code

QDomNodeList roles=login.elementsByTagName(role\_text);

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

{

QDomNode rolenode = roles.at(i);

if(rolenode.isElement())

{

QDomElement role = rolenode.toElement();

if(ListElements(role,uname,pword))

return 1;

}

}

return 0;

}

#endif // CHECKLOGIN\_H

# 4.CheckLogin.h:

#ifndef CHECKMEDICINES\_H

#define CHECKMEDICINES\_H

#include <QDialog>

#include <QtCore>

#include <QtXml>

#include <QDebug>

class CheckMedicineClass

{

public:

QString quantity\_available,mrp,medname,productid,location;

bool CheckMedicine(QString medname);

};

bool CheckMedicineClass::CheckMedicine(QString medname)

{

this->medname=medname;

quantity\_available=0;

mrp=0;

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

throw -1;

}

if(!document.setContent(&file))

{

file.close();

throw -2;

}

QDomElement medicines = document.firstChildElement();

QDomNodeList medicineslist = medicines.elementsByTagNme("medicine");

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

{

QDomNode medicinenode = medicineslist.at(i);

if(medicinenode.isElement())

{

QDomElement medicineele=medicinenode.toElement();

if(medname==medicineele.attribute("name"))

{

productid=medicineele.attribute("id");

quantity\_available=medicineele.attribute("quantity");

mrp=medicineele.attribute("mrp");

location=medicineele.attribute("location");

return 1;

}

}

}

file.close();

return 0;

}

#endif // CHECKMEDICINES\_H

# 5.EditMedicineList.h:

#ifndef EDITMEDICINELIST\_H

#define EDITMEDICINELIST\_H

#include <QtCore>

#include <QtXml>

#endif // EDITMEDICINELIST\_H

# 6.expirydates.h:

#ifndef EXPIRYDATES\_H

#define EXPIRYDATES\_H

#include <QDialog>

namespace Ui {

class ExpiryDates;

}

class ExpiryDates : public QDialog

{

Q\_OBJECT

public:

explicit ExpiryDates(QWidget \*parent = nullptr);

~ExpiryDates();

private slots:

void check();

//void supplier();

void on\_PlaceOrderButton\_clicked();

private:

Ui::ExpiryDates \*ui;

};

#endif // EXPIRYDATES\_H

**7.mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "showmenu.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

ShowMenu \*myShowMenu;

~MainWindow();

private slots:

void on\_loginbutton\_clicked();

void on\_showhide\_button\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**8.placeorderdialog.h:**

#ifndef PLACEORDERDIALOG\_H

#define PLACEORDERDIALOG\_H

#include <QDialog>

namespace Ui {

class PlaceOrderDialog;

}

class PlaceOrderDialog : public QDialog

{

Q\_OBJECT

public:

explicit PlaceOrderDialog(QWidget \*parent = nullptr);

~PlaceOrderDialog();

private slots:

void DisplayPlaceOrderTable();

void on\_pushButton\_clicked();

void on\_OrderRecievedButton\_clicked();

private:

Ui::PlaceOrderDialog \*ui;

};

#endif // PLACEORDERDIALOG\_H

# 9.showinvoice.h:

#ifndef SHOWINVOICE\_H

#define SHOWINVOICE\_H

#include <QDialog>

namespace Ui {

class ShowInvoice;

}

class ShowInvoice : public QDialog

{

Q\_OBJECT

public:

explicit ShowInvoice(QWidget \*parent = nullptr);

~ShowInvoice();

private slots:

void WriteInInvoice();

void on\_pushButton\_clicked();

private:

Ui::ShowInvoice \*ui;

};

#endif // SHOWINVOICE\_H

**10.ShowMedList.h:**

#ifndef SHOWMEDLIST\_H

#define SHOWMEDLIST\_H

#include <QDialog>

namespace Ui {

class ShowMedList;

}

class ShowMedList : public QDialog

{

Q\_OBJECT

public:

explicit ShowMedList(QWidget \*parent = nullptr);

~ShowMedList();

private slots:

void on\_AddnewmedButton\_clicked();

void on\_DeleteMedButton\_clicked();

//void on\_pushButton\_2\_clicked();

void on\_EditButton\_clicked();

private:

Ui::ShowMedList \*ui;

};

#endif // SHOWMEDLIST\_H

**11.showmenu.h:**

#ifndef SHOWMENU\_H

#define SHOWMENU\_H

#include "buymeddialog.h"

#include "showtransactions.h"

#include "expirydates.h"

#include <ShowMedList.h>

#include "supplierdata.h"

#include "placeorderdialog.h"

#include <QWidget>

namespace Ui {

class ShowMenu;

}

class ShowMenu : public QWidget

{

Q\_OBJECT

public:

explicit ShowMenu(QString Cashier,QWidget \*parent = nullptr);

BuyMedDialog \*myBuyMedDialog;

ShowTransactions \*myShowTransactions;

ExpiryDates \*myExpiryDates;

ShowMedList \*myShowMedList;

supplierData \*mysupplierData;

PlaceOrderDialog \*myPlaceOrderDialog;

QString Cashier;

~ShowMenu();

private slots:

void on\_BillingButton\_clicked();

void on\_ExpiryButton\_clicked();

void on\_PlaceOrderButton\_clicked();

void on\_FilesButton\_clicked();

void on\_MedFileButton\_clicked();

void on\_SupplierFileButton\_clicked();

void on\_TransactionFileButton\_clicked();

private:

Ui::ShowMenu \*ui;

};

#endif // SHOWMENU\_H

**12.showtransactions.h:**

#ifndef SHOWTRANSACTIONS\_H

#define SHOWTRANSACTIONS\_H

#include <QDialog>

namespace Ui {

class ShowTransactions;

}

class ShowTransactions : public QDialog

{

Q\_OBJECT

public:

explicit ShowTransactions(QWidget \*parent = nullptr);

~ShowTransactions();

private slots:

void WriteTransactions();

private:

Ui::ShowTransactions \*ui;

};

#endif // SHOWTRANSACTIONS\_H

# 13.supplierdata.h:

#ifndef SUPPLIERDATA\_H

#define SUPPLIERDATA\_H

#include <QDialog>

namespace Ui {

class supplierData;

}

class supplierData : public QDialog

{

Q\_OBJECT

public:

explicit supplierData(QWidget \*parent = nullptr);

~supplierData();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_3\_clicked();

private:

Ui::supplierData \*ui;

};

#endif // SUPPLIERDATA\_H

**b. sources:**

**1.** **addmed.cpp:**

#include "addmed.h"

#include "ui\_addmed.h"

#include <QtCore>

#include <QtXml>

#include <ShowMedList.h>

Addmed::Addmed(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Addmed)

{

ui->setupUi(this);

}

Addmed::~Addmed()

{

delete ui;

}

void Addmed::on\_pushButton\_clicked()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug () << "Error saving XML file....";

file.close();

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

return ;

file.close();

}

QDomElement medicines = document.documentElement();

file.close();

QDomElement medicine=document.createElement("medicine");

medicine.setAttribute("id",ui->lineEdit->text());

medicine.setAttribute("name",ui->lineEdit\_2->text());

medicine.setAttribute("quantity",ui->lineEdit\_3->text());

medicine.setAttribute("mrp",ui->lineEdit\_4->text());

medicine.setAttribute("dd",ui->lineEdit\_5->text());

medicine.setAttribute("mm",ui->lineEdit\_6->text());

medicine.setAttribute("yy",ui->lineEdit\_7->text());

medicine.setAttribute("status",ui->lineEdit\_8->text());

medicine.setAttribute("location",ui->lineEdit\_9->text());

medicines.appendChild(medicine);

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

hide();

}

**2.** **buymeddialog.cpp:**

#include "buymeddialog.h"

#include "ui\_buymeddialog.h"

#include "checkmedicines.h"

#include <QApplication>

#include <QMessageBox>

#include <ctime>

#include <string.h>

CheckMedicineClass obj;

BuyMedDialog::BuyMedDialog(QString Cashier,QWidget \*parent)

: QDialog(parent)

, ui(new Ui::BuyMedDialog)

{

ui->setupUi(this);

this->Cashier=Cashier;

setWindowTitle("SRM MEDICAL STORE");

ui->medicine\_lineedit->setClearButtonEnabled(true);

ui->allitems\_table->horizontalHeader()->setDefaultSectionSize(125);

ui->allitems\_table->horizontalHeader()->setStretchLastSection(true);

ui->allitems\_table->horizontalHeader()->setVisible(true);

}

BuyMedDialog::~BuyMedDialog()

{

delete ui;

}

void BuyMedDialog::on\_generatebill\_button\_clicked()

{

if(ui->show\_tvalue->text()=="0" || ui->custname\_linedit->text()=="" || ui->custid\_lineedit->text()=="")

{

QMessageBox::information(this,"Error","Enter required information to generate the bill");

return;

}

WriteTransactionXML();

EditMedicineList();

myShowInvoice=new ShowInvoice;

myShowInvoice->show();

}

void BuyMedDialog::on\_addbutton\_clicked()

{

if((obj.quantity\_available).toInt()==0)

{

QMessageBox::information(this,"Error","Cannot add an unavailable product ");

return;

}

if(ui->quantity\_sb->value()==0 || ui->quantity\_sb->value()>=obj.quantity\_available.toInt())

{

QMessageBox::information(this,"Error","Cannot add an product with unavailable quantity !");

return;

}

int quantity\_entered,mrp,total\_value;

total\_value=(ui->show\_tvalue->text()).toInt();

qDebug()<<total\_value;

quantity\_entered= ui->quantity\_sb->value();

mrp=(ui->mrp\_label->text()).toInt();

ui->allitems\_table->insertRow(ui->allitems\_table->rowCount());

QTableWidgetItem \* item;

for(int j=0;j<5;j++)

{

item = new QTableWidgetItem;

if(j==0)

item->setText(obj.productid);

if(j==1)

item->setText(ui->medicine\_lineedit->text());

if(j==2)

item->setText(QString::number(quantity\_entered));

if(j==3)

item->setText(QString::number(mrp));

if(j==4)

item->setText(QString::number(mrp\*quantity\_entered));

ui->allitems\_table->setItem(ui->allitems\_table->rowCount()-1,j,item);

}

total\_value+=mrp\*quantity\_entered;

ui->show\_tvalue->setText(QString::number(total\_value));

ui->t\_items\_label->setText(QString::number(ui->allitems\_table->rowCount()));

ui->medicine\_lineedit->setText("");

ui->avai\_quant\_label->setText("");

ui->mrp\_label->setText("");

ui->quantity\_sb->setValue(0);

}

void BuyMedDialog::on\_search\_button\_clicked()

{

QString medname=ui->medicine\_lineedit->text();

if(medname=="")

{

QMessageBox::information(this,"Error","Enter the product name to search !");

return;

}

try

{

if(obj.CheckMedicine(medname))

{

if((obj.quantity\_available).toInt()==0)

ui->avai\_quant\_label->setText("NOT AVAILABLE");

else

ui->avai\_quant\_label->setText(obj.quantity\_available);

ui->LocationLabel->setText(obj.location);

ui->mrp\_label->setText(obj.mrp);

}

else

{

ui->avai\_quant\_label->setText("NOT AVAILABLE");

ui->mrp\_label->setText("");

ui->quantity\_sb->setValue(0);

}

}

catch(int error)

{

if(error==-1){QMessageBox::information(this,"Error","Failed to open file");}

else{QMessageBox::information(this,"Error","Failed to load document");}

}

}

void BuyMedDialog::on\_delete\_button\_clicked()

{

auto rowList = ui->allitems\_table->selectionModel()->selectedIndexes();

if(rowList.empty())

{

QMessageBox::information(this,"Error","Select an item to delete");

return;

}

int total\_value,value\_tobe\_removed;

total\_value=(ui->show\_tvalue->text()).toInt();

value\_tobe\_removed=(ui->allitems\_table->item(rowList.constFirst().row(),4)->text()).toInt();

ui->allitems\_table->removeRow(rowList.constFirst().row());

total\_value-=value\_tobe\_removed;

ui->show\_tvalue->setText(QString::number(total\_value));

ui->t\_items\_label->setText(QString::number(ui->t\_items\_label->text().toInt()-1));

}

void BuyMedDialog::on\_cancel\_button\_clicked()

{

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Confirm Cancel", "Cancel The Billing?",QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes)

{

this->close();

}

}

void BuyMedDialog::WriteTransactionXML()

{

time\_t now=time(0);

tm \*ltm=localtime(&now);

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\transactions.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QMessageBox::information(this,"Error","Failed to open file");

return;

}

if(!document.setContent(&file))

{

QMessageBox::information(this,"Error","Failed to load document");

file.close();

return ;

}

QDomElement Transactions = document.documentElement();

file.close();

QDomElement Transaction=document.createElement("Transaction");

Transaction.setAttribute("BillNo","");

Transaction.setAttribute("Date",QString::number(ltm->tm\_mday)+"/"+QString::number(ltm->tm\_mon + 1)+"/"+QString::number(ltm->tm\_year + 1900));

Transaction.setAttribute("Time",QString::number(ltm->tm\_hour)+":"+ QString::number(ltm->tm\_min)+":"+QString::number(ltm->tm\_sec));

Transaction.setAttribute("Cashier",Cashier);

Transactions.appendChild(Transaction);

QDomElement Customer=document.createElement("Customer");

Customer.setAttribute("CustomerName",ui->custname\_linedit->text());

Customer.setAttribute("CustomerID",ui->custid\_lineedit->text());

Transaction.appendChild(Customer);

QDomElement Orders=document.createElement("Orders");

Orders.setAttribute("TotalValue",ui->show\_tvalue->text());

Transaction.appendChild(Orders);

QDomElement Order;

for(int i=0;i<ui->allitems\_table->rowCount();i++)

{

Order= document.createElement("Order");

Order.setAttribute("ProductID",ui->allitems\_table->item(i,0)->text());

Order.setAttribute("ProductName",ui->allitems\_table->item(i,1)->text());

Order.setAttribute("Quantity",ui->allitems\_table->item(i,2)->text());

Order.setAttribute("Mrp",ui->allitems\_table->item(i,3)->text());

Orders.appendChild(Order);

}

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

}

void BuyMedDialog::EditMedicineList()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug () << "Error saving XML file....";

file.close();

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

return ;

file.close();

}

file.close();

QDomElement medicines = document.documentElement();

QDomNodeList medicine = medicines.childNodes();

QString id,EditedQuantity,QuantityToRemove;

for(int j=0;j<(ui->t\_items\_label->text()).toInt();j++)

{

id=ui->allitems\_table->item(j,0)->text();

QuantityToRemove=ui->allitems\_table->item(j,2)->text();

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

{

QDomNode medicinenode = medicine.at(i);

if (medicinenode.isElement())

{

QDomElement medicineele=medicinenode.toElement();

if(medicineele.attribute("id")==id)

{

EditedQuantity=QString::number(medicineele.attribute("quantity").toInt()-QuantityToRemove.toInt());

medicineele.setAttribute("quantity",EditedQuantity);

}

}

}

}

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

}

# 3.expirydates.cpp:

#include "expirydates.h"

#include "ui\_expirydates.h"

#include <ctime>

#include <QtCore>

#include <QtXml>

#include <difference\_between\_date.h>

ExpiryDates::ExpiryDates(QWidget \*parent) :

QDialog(parent),

ui(new Ui::ExpiryDates)

{

ui->setupUi(this);

ui->ExpiryTable->horizontalHeader()->setDefaultSectionSize(120);

ui->ExpiryTable->horizontalHeader()->setStretchLastSection(true);

ui->ExpiryTable->horizontalHeader()->setVisible(true);

check();

}

ExpiryDates::~ExpiryDates()

{

delete ui;

}

void ExpiryDates::check()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return;

}

QDomElement medicines = document.firstChildElement();

QDomNodeList medicineslist1 = medicines.elementsByTagName("medicine");

time\_t now = time(0);

tm \*ltm = localtime(&now);

for(int i=0;i<medicineslist1.count();i++){

QDomNode medicinenode = medicineslist1.at(i);

if(medicinenode.isElement())

{

QDomElement medicineele=medicinenode.toElement();

Date d1,d2;

d1={ltm->tm\_mday,1 + ltm->tm\_mon,1900 + ltm->tm\_year};

d2={(medicineele.attribute("dd")).toInt(),(medicineele.attribute("mm")).toInt(),(medicineele.attribute("yy")).toInt()};

if(getDifference(d1, d2)<=30)

{

ui->ExpiryTable->insertRow(ui->ExpiryTable->rowCount());

QTableWidgetItem \* item;

for(int j=0;j<=6;j++)

{

item = new QTableWidgetItem;

if(j==0 )

item->setText(medicineele.attribute("id"));

if(j==1 )

item->setText(medicineele.attribute("name"));

if(j==2 )

item->setText(medicineele.attribute("location"));

if(j==3 )

item->setText(medicineele.attribute("quantity"));

if(j==4 )

item->setText(medicineele.attribute("mrp"));

if(j==5 )

item->setText(medicineele.attribute("dd")+"-"+medicineele.attribute("mm")+"-"+medicineele.attribute("yy"));

if(j==6)

item->setText(medicineele.attribute("status"));

ui->ExpiryTable->setItem(ui->ExpiryTable->rowCount()-1,j,item);

}

}

}

}

}

/\*

void ExpiryDates::supplier()

{

QDomDocument document;

QFile file("C:\\Users\\91868\\OneDrive\\Desktop\\cpp\\Supplier.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return ;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return;

}

file.close();

QDomElement supply = document.firstChildElement();

QDomNodeList supplierlist = supply.elementsByTagName("supplier");

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

{

QDomNode suppliernode = supplierlist.at(i);

if(suppliernode.isElement())

{

QDomElement supplierele=suppliernode.toElement();

supplierele.

}

}

}

\*/

void ExpiryDates::on\_PlaceOrderButton\_clicked()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");\

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return ;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return ;

}

file.close();

QDomElement medicines = document.firstChildElement();

auto rowList = ui->ExpiryTable->selectionModel()->selectedIndexes();

QString id=(ui->ExpiryTable->item(rowList.constFirst().row(),0)->text());

QDomNodeList medicineslist1 = medicines.elementsByTagName("medicine");

for(int i=0;i<medicineslist1.count();i++){

QDomNode medicinenode = medicineslist1.at(i);

if(medicinenode.isElement()){

QDomElement medicineele=medicinenode.toElement();

if(id==medicineele.attribute("id"))

{

medicineele.setAttribute("status","YES");

ui->ExpiryTable->item(rowList.constFirst().row(),6)->setText("YES");

}

}

}

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

}

**4.main.cpp:**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

**5.mainwindow.cpp:**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "CheckLogin.h"

#include <QApplication>

#include <QMessageBox>

#include <iostream>

#include <string.h>

using namespace std;

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->showMaximized();

ui->verticalLayout\_2->setAlignment(Qt::AlignCenter);

setWindowTitle("SRM MEDICAL STORE");

ui->getusername->setClearButtonEnabled(true);

ui->getpassword->setClearButtonEnabled(true);

ui->getpassword->setEchoMode(QLineEdit::Password);

ui->showhide\_button->setIcon(QIcon("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\hidepng.png"));

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_loginbutton\_clicked()

{

QString username = ui->getusername->text();

QString password = ui->getpassword->text();

QString role\_text = ui->getrole->currentText();

if(CheckLogin(role\_text,username,password)==1)

{

this->close();

myShowMenu=new ShowMenu(ui->getusername->text());

myShowMenu->show();

}

else

{

QMessageBox::information(this,"login","Incorrect password or username or role !");

}

}

void MainWindow::on\_showhide\_button\_clicked()

{

static bool pressed;

if(!pressed)

{

ui->getpassword->setEchoMode(QLineEdit::Normal);

ui->showhide\_button->setIcon(QIcon("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\showpng.png"));

pressed=1;

}

else

{

ui->getpassword->setEchoMode(QLineEdit::Password);

ui->showhide\_button->setIcon(QIcon("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\hidepng.png"));

pressed=0;

}

}

**6.placeorderdialog.cpp:**

#include "placeorderdialog.h"

#include "ui\_placeorderdialog.h"

#include<addmed.h>

#include <QtCore>

#include <QtXml>

PlaceOrderDialog::PlaceOrderDialog(QWidget \*parent) :

QDialog(parent),

ui(new Ui::PlaceOrderDialog)

{

ui->setupUi(this);

ui->PlaceOrderTable->horizontalHeader()->setDefaultSectionSize(120);

ui->PlaceOrderTable->horizontalHeader()->setStretchLastSection(true);

ui->PlaceOrderTable->horizontalHeader()->setVisible(true);

DisplayPlaceOrderTable();

}

PlaceOrderDialog::~PlaceOrderDialog()

{

delete ui;

}

void PlaceOrderDialog::on\_pushButton\_clicked()

{

Addmed x;

x.setModal(true);

x.exec();

}

void PlaceOrderDialog::on\_OrderRecievedButton\_clicked()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return;

}

file.close();

QDomElement medicines = document.firstChildElement();

QString value;

auto rowList = ui->PlaceOrderTable->selectionModel()->selectedIndexes();

value=(ui->PlaceOrderTable->item(rowList.constFirst().row(),0)->text());

QDomNodeList medicineslist1 = medicines.elementsByTagName("medicine");

for(int i=0;i<medicineslist1.count();i++){

QDomNode medicinenode = medicineslist1.at(i);

if(medicinenode.isElement()){

QDomElement medicineele=medicinenode.toElement();

if(value==medicineele.attribute("id"))

{

medicines.setAttribute("status","NO");

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

ui->PlaceOrderTable->item(rowList.constFirst().row(),4)->setText("NO");

}

}

}

}

void PlaceOrderDialog::DisplayPlaceOrderTable()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return ;

}

QDomElement medicines = document.firstChildElement();

QDomNodeList medicineslist = medicines.elementsByTagName("medicine");

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

QDomNode medicinenode = medicineslist.at(i);

if(medicinenode.isElement())

{

QDomElement medicineele=medicinenode.toElement();

if((medicineele.attribute("status"))=="YES")

{

QTableWidgetItem \* item;

ui->PlaceOrderTable->insertRow(ui->PlaceOrderTable->rowCount());

for(int j=0;j<=4;j++)

{

item = new QTableWidgetItem;

if(j==0 )

item->setText(medicineele.attribute("id"));

if(j==1 )

item->setText(medicineele.attribute("name"));

if(j==2 )

item->setText(medicineele.attribute("quantity"));

if(j==3 )

item->setText(medicineele.attribute("mrp"));

if(j==4 ){

item->setText(medicineele.attribute("status"));

}

ui->PlaceOrderTable->setItem(ui->PlaceOrderTable->rowCount()-1,j,item);

}

}

}

}

}

/\*

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

{

}

\*/

**7.showinvoice.cpp:**

#include "showinvoice.h"

#include "C:\Users\Durga Vinay\Desktop\Cpp project\build-billmaking-Desktop\_Qt\_6\_4\_1\_MinGW\_64\_bit-Debug\ui\_showinvoice.h"

#include <QDomDocument>

#include <QtXml>

#include <QMessageBox>

ShowInvoice::ShowInvoice(QWidget \*parent) :

QDialog(parent),

ui(new Ui::ShowInvoice)

{

ui->setupUi(this);

ui->allitems\_table->horizontalHeader()->setDefaultSectionSize(120);

ui->allitems\_table->horizontalHeader()->setStretchLastSection(true);

ui->allitems\_table->horizontalHeader()->setVisible(true);

WriteInInvoice();

}

ShowInvoice::~ShowInvoice()

{

delete ui;

}

void ShowInvoice::WriteInInvoice()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\transactions.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QMessageBox::information(this,"Error","Failed to open file");

return;

}

if(!document.setContent(&file))

{

QMessageBox::information(this,"Error","Failed to load document");

file.close();

return;

}

QDomElement Transactions=document.documentElement();

QDomElement Transaction=Transactions.lastChildElement();

QDomNodeList TrasactionsList = Transactions.elementsByTagName("Transaction");

Transaction.setAttribute("BillNo",TrasactionsList.count());

QDomElement Customer=Transaction.firstChildElement();

QDomElement Orders=Transaction.lastChildElement();

QDomNodeList OrdersList = Orders.elementsByTagName("Order");

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

{

QDomNode OrderNode = OrdersList.at(i);

if(OrderNode.isElement())

{

QDomElement Order=OrderNode.toElement();

ui->allitems\_table->insertRow(ui->allitems\_table->rowCount());

QTableWidgetItem \* item;

QString mrp,quantity;

for(int j=0;j<4;j++)

{

item = new QTableWidgetItem;

mrp=Order.attribute("Mrp");

quantity=Order.attribute("Quantity");

if(j==0)

item->setText(Order.attribute("ProductName"));

if(j==1)

item->setText(quantity);

if(j==2)

item->setText(mrp);

if(j==3)

item->setText(QString::number(mrp.toInt()\*quantity.toInt()));

ui->allitems\_table->setItem(i,j,item);

}

}

}

ui->time\_label->setText(Transaction.attribute("Time"));

ui->date\_label->setText(Transaction.attribute("Date"));

ui->billno\_label->setText(Transaction.attribute("BillNo"));

ui->show\_c\_name->setText(Customer.attribute("CustomerName"));

ui->show\_c\_id->setText(Customer.attribute("CustomerID"));

ui->t\_value\_label->setText(Orders.attribute("TotalValue"));

ui->cashier\_label->setText(Transaction.attribute("Cashier"));

file.close();

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

return ;

}

void ShowInvoice::on\_pushButton\_clicked()

{

this->close();

}

**8.showmedlist.cpp:**

#include "ShowMedList.h"

#include "ui\_ShowMedList.h"

#include <QtCore>

#include <QtXml>

#include<addmed.h>

ShowMedList::ShowMedList(QWidget \*parent) :

QDialog(parent),

ui(new Ui::ShowMedList)

{

ui->setupUi(this);

ui->tableWidget->horizontalHeader()->setDefaultSectionSize(120);

ui->tableWidget->horizontalHeader()->setStretchLastSection(true);

ui->tableWidget->horizontalHeader()->setVisible(true);

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return;

}

file.close();

QDomElement medicines = document.firstChildElement();

QDomNodeList medicineslist1 = medicines.elementsByTagName("medicine");

for(int i=0;i<medicineslist1.count();i++){

QDomNode medicinenode = medicineslist1.at(i);

if(medicinenode.isElement())

{

QDomElement medicineele=medicinenode.toElement();

ui->tableWidget->insertRow(ui->tableWidget->rowCount());

QTableWidgetItem \* item;

for(int j=0;j<=6;j++)

{

item = new QTableWidgetItem;

if(j==0 )

item->setText(medicineele.attribute("id"));

if(j==1 )

item->setText(medicineele.attribute("name"));

if(j==2 )

item->setText(medicineele.attribute("quantity"));

if(j==3 )

item->setText(medicineele.attribute("mrp"));

if(j==4 ){

item->setText(medicineele.attribute("dd")+"-"+medicineele.attribute("mm")+"-"+medicineele.attribute("yy"));

}

if(j==5)

item->setText(medicineele.attribute("location"));

if(j==6 )

item->setText(medicineele.attribute("status"));

ui->tableWidget->setItem(ui->tableWidget->rowCount()-1,j,item);

}

}

}

}

ShowMedList::~ShowMedList()

{

delete ui;

}

void ShowMedList::on\_AddnewmedButton\_clicked()

{

Addmed x;

x.setModal(true);

x.exec();

if(QWidget::isVisible())

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return;

}

file.close();

QDomElement medicines=document.documentElement();

QDomElement medicine=medicines.lastChildElement();

ui->tableWidget->insertRow(ui->tableWidget->rowCount());

QTableWidgetItem \* item;

for(int j=0;j<=6;j++)

{

item = new QTableWidgetItem;

if(j==0 )

item->setText(medicine.attribute("id"));

if(j==1 )

item->setText(medicine.attribute("name"));

if(j==2 )

item->setText(medicine.attribute("quantity"));

if(j==3 )

item->setText(medicine.attribute("mrp"));

if(j==4 )

item->setText(medicine.attribute("dd")+"-"+medicine.attribute("mm")+"-"+medicine.attribute("yy"));

ui->tableWidget->setItem(ui->tableWidget->rowCount()-1,j,item);

}

}

}

void ShowMedList::on\_DeleteMedButton\_clicked()

{

QString value\_tobe\_removed;

auto rowList = ui->tableWidget->selectionModel()->selectedIndexes();

value\_tobe\_removed=(ui->tableWidget->item(rowList.constFirst().row(),0)->text());

ui->tableWidget->removeRow(rowList.constFirst().row());

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"showmedlist Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Transactions read Failed to load document";

return;

file.close();

}

file.close();

QDomElement medicines = document.documentElement();

QDomNodeList medicine = medicines.childNodes();

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

QDomNode node = medicine.at(i);

if (node.isElement() && node.toElement().attribute("id") == value\_tobe\_removed) {

medicines.removeChild(node);

}

}

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

}

void ShowMedList::on\_EditButton\_clicked()

{

auto SelectedList = ui->tableWidget->selectionModel()->selectedIndexes();

if(SelectedList.empty())

{

ui->tableWidget->setSelectionBehavior(QAbstractItemView::SelectItems);

ui->EditButton->setText("Save");

}

else

{

QString value\_to\_changed=ui->tableWidget->item(SelectedList.constFirst().row(),SelectedList.constFirst().column())->text();

QString attribute\_to\_be\_changed=ui->tableWidget->horizontalHeaderItem(SelectedList.constFirst().column())->text();

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\medicineslist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"showmedlist Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

return;

file.close();

}

file.close();

QDomElement medicines = document.documentElement();

QDomNodeList medicine = medicines.childNodes();

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

{

QDomNode medicinenode = medicine.at(i);

if (medicinenode.isElement())

{

QDomElement medicineele=medicinenode.toElement();

if(medicineele.attribute("id")==ui->tableWidget->item(SelectedList.constFirst().row(),0)->text())

{

qDebug()<<medicineele.attribute("name");

medicineele.setAttribute(attribute\_to\_be\_changed,value\_to\_changed);

}

}

}

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

ui->EditButton->setText("Edit");

ui->tableWidget->setSelectionBehavior(QAbstractItemView::SelectRows);

}

}

**9.showmenu.cpp:**

#include "showmenu.h"

#include "ui\_showmenu.h"

#include <qpixmap.h>

ShowMenu::ShowMenu(QString Cashier,QWidget \*parent) :

QWidget(parent),

ui(new Ui::ShowMenu)

{

ui->setupUi(this);

this->showMaximized();

this->Cashier=Cashier;

ui->AllFilesButtons->hide();

QPixmap pix("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\SRMap logo.png");

ui->SRMlogo->setPixmap(pix.scaled(100,100,Qt::KeepAspectRatioByExpanding));

}

ShowMenu::~ShowMenu()

{

delete ui;

}

void ShowMenu::on\_BillingButton\_clicked()

{

myBuyMedDialog = new BuyMedDialog(Cashier);

myBuyMedDialog->show();

}

void ShowMenu::on\_ExpiryButton\_clicked()

{

myExpiryDates = new ExpiryDates;

myExpiryDates->show();

}

void ShowMenu::on\_PlaceOrderButton\_clicked()

{

myPlaceOrderDialog = new PlaceOrderDialog;

myPlaceOrderDialog->show();

}

void ShowMenu::on\_FilesButton\_clicked()

{

if(ui->AllFilesButtons->isVisible())

ui->AllFilesButtons->hide();

else

ui->AllFilesButtons->show();

}

void ShowMenu::on\_MedFileButton\_clicked()

{

myShowMedList = new ShowMedList;

myShowMedList->show();

}

void ShowMenu::on\_SupplierFileButton\_clicked()

{

mysupplierData = new supplierData;

mysupplierData->show();

}

void ShowMenu::on\_TransactionFileButton\_clicked()

{

myShowTransactions = new ShowTransactions;

myShowTransactions->show();

}

**10.showtransactions.cpp:**

#include "showtransactions.h"

#include "ui\_showtransactions.h"

#include <QDomDocument>

#include <QtXml>

#include <QMessageBox>

ShowTransactions::ShowTransactions(QWidget \*parent) :

QDialog(parent),

ui(new Ui::ShowTransactions)

{

ui->setupUi(this);

ui->TransactionsTable->horizontalHeader()->setDefaultSectionSize(120);

ui->TransactionsTable->horizontalHeader()->setStretchLastSection(true);

ui->TransactionsTable->horizontalHeader()->setVisible(true);

WriteTransactions();

}

ShowTransactions::~ShowTransactions()

{

delete ui;

}

void ShowTransactions::WriteTransactions()

{

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\transactions.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QMessageBox::information(this,"Error","Failed to open file");

return;

}

if(!document.setContent(&file))

{

QMessageBox::information(this,"Error","Failed to load document");

file.close();

return;

}

QDomElement Transactions=document.documentElement();

QDomNodeList TrasactionsList = Transactions.elementsByTagName("Transaction");

int TransactionsCount = TrasactionsList.count();

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

{

QDomNode TransactionNode = TrasactionsList.at(TransactionsCount-i);

if(TransactionNode.isElement())

{

QDomElement Transaction=TransactionNode.toElement();

QDomElement Customer = Transaction.firstChildElement();

QDomElement Orders=Transaction.lastChildElement();

QDomNodeList OrdersList = Orders.elementsByTagName("Order");

int rowCount = ui->TransactionsTable->rowCount();

ui->TransactionsTable->insertRow(rowCount);

QTableWidgetItem \* item;

for(int j=0;j<10;j++)

{

item = new QTableWidgetItem;

if(j==0)

item->setText(Transaction.attribute("BillNo"));

if(j==1)

item->setText(Transaction.attribute("Date"));

if(j==2)

item->setText(Transaction.attribute("Time"));

if(j==3)

item->setText(Orders.attribute("TotalValue"));

if(j==4)

item->setText(Customer.attribute("CustomerID"));

if(j==5)

item->setText(Customer.attribute("CustomerName"));

if(j==6)

item->setText(Transaction.attribute("Cashier"));

if(j==7)

item->setText(Orders.firstChildElement().attribute("ProductID"));

if(j==8)

item->setText(Orders.firstChildElement().attribute("ProductName"));

if(j==9)

item->setText(Orders.firstChildElement().attribute("Quantity"));

ui->TransactionsTable->setItem(rowCount,j,item);

}

for(int i=1;i<OrdersList.count();i++)

{

QDomNode OrderNode = OrdersList.at(i);

if(OrderNode.isElement())

{

QDomElement Order=OrderNode.toElement();

ui->TransactionsTable->insertRow(rowCount+i);

QTableWidgetItem \* item;

for(int j=7;j<=9;j++)

{

item = new QTableWidgetItem;

if(j==7)

item->setText(Order.attribute("ProductID"));

if(j==8)

item->setText(Order.attribute("ProductName"));

if(j==9)

item->setText(Order.attribute("Quantity"));

ui->TransactionsTable->setItem(rowCount+i,j,item);

}

}

}

}

}

file.close();

return ;

}

**11.supplierdata.cpp:**

#include "supplierdata.h"

#include "ui\_supplierdata.h"

#include <QtCore>

#include <QtXml>

//#include<addsupplier.h>

supplierData::supplierData(QWidget \*parent) :

QDialog(parent),

ui(new Ui::supplierData)

{

ui->setupUi(this);

ui->tableWidget->horizontalHeader()->setDefaultSectionSize(150);

ui->tableWidget->horizontalHeader()->setStretchLastSection(true);

ui->tableWidget->horizontalHeader()->setVisible(true);

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\Supplierlist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return ;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

file.close();

return ;

}

QDomElement suppliers = document.firstChildElement();

QDomNodeList Supplier = suppliers.elementsByTagName("supplier");

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

{

QDomNode suppliernode = Supplier.at(i);

if(suppliernode.isElement())

{

QDomElement supplierele=suppliernode.toElement();

QDomElement Medicines=supplierele.firstChildElement();

QDomNodeList MedicinesList = Medicines.elementsByTagName("medicine");

int rowCount = ui->tableWidget->rowCount();

ui->tableWidget->insertRow(rowCount);

QTableWidgetItem \* item;

qDebug()<<Medicines.firstChildElement().attribute("id");

for(int j=0;j<=4;j++)

{

item = new QTableWidgetItem;

if(j==0 )

item->setText(supplierele.attribute("Name"));

if(j==1 )

item->setText(supplierele.attribute("company"));

if(j==2)

item->setText(Medicines.firstChildElement().attribute("id"));

if(j==3)

item->setText(Medicines.firstChildElement().attribute("name"));

if(j==4 )

item->setText(supplierele.attribute("contact"));

ui->tableWidget->setItem(rowCount,j,item);

}

for(int i=1;i<MedicinesList.count();i++)

{

QDomNode MedicineNode = MedicinesList.at(i);

if(MedicineNode.isElement())

{

QDomElement Medicine=MedicineNode.toElement();

ui->tableWidget->insertRow(rowCount+i);

QTableWidgetItem \* item;

for(int j=2;j<=3;j++)

{

item = new QTableWidgetItem;

if(j==2)

item->setText(Medicine.attribute("id"));

if(j==3)

item->setText(Medicine.attribute("name"));

ui->tableWidget->setItem(rowCount+i,j,item);

}

}

}

}

}

file.close();

}

supplierData::~supplierData()

{

delete ui;

}

void supplierData::on\_pushButton\_clicked()

{/\*

Addsupplier x;

x.setModal(true);

x.exec();\*/

//if(QWidget::isVisible()){

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\Supplierlist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Failed to load document";

return;

file.close();

}

QDomElement suppliers=document.documentElement();

QDomElement supplier=suppliers.lastChildElement();

ui->tableWidget->insertRow(ui->tableWidget->rowCount());

QTableWidgetItem \* item;

for(int j=0;j<=6;j++)

{

item = new QTableWidgetItem;

if(j==0 )

item->setText(supplier.attribute("Name"));

if(j==1 )

item->setText(supplier.attribute("company"));

if(j==2 )

item->setText(supplier.attribute("med\_name"));

if(j==3 )

item->setText(supplier.attribute("contact"));

ui->tableWidget->setItem(ui->tableWidget->rowCount()-1,j,item);

}

file.close();

//}

}

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

{

QString value\_tobe\_removed;

auto rowList = ui->tableWidget->selectionModel()->selectedIndexes();

value\_tobe\_removed=(ui->tableWidget->item(rowList.constFirst().row(),0)->text());

ui->tableWidget->removeRow(rowList.constFirst().row());

QDomDocument document;

QFile file("C:\\Users\\Durga Vinay\\Desktop\\Cpp project\\XML files\\Supplierlist.xml");

if(!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

qDebug()<<"Failed to open file";

return;

}

if(!document.setContent(&file))

{

qDebug()<<"Transactions read Failed to load document";

return;

file.close();

}

file.close();

QDomElement suppliers = document.documentElement();

QDomNodeList supplier = suppliers.childNodes();

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

QDomNode node = supplier.at(i);

if (node.isElement() && node.toElement().attribute("Name") == value\_tobe\_removed) {

suppliers.removeChild(node);

}

}

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream output(&file);

output << document.toString();

file.close();

}

**XML Files:**

**MedicineList.XML:**

<medicines>

 <medicine ID="4" location="A1" yy="2023" quantity="83" status="YES" mm="12" mrp="2" name="dolo 650" id="0" dd="22" MRP="5"/>

 <medicine ID="5" location="B1" yy="2022" quantity="200" status="YES" mm="12" mrp="3" name="Probiotics" id="1" dd="12"/>

 <medicine ID="6" location="C1" yy="2023" quantity="300" status="NO" mm="12" mrp="4" name="Vitamin C" id="2" dd="20"/>

 <medicine ID="7" location="D1" yy="2022" quantity="250" status="NO" mm="10" mrp="8" name="Acid Free C" id="3" dd="20"/>

 <medicine ID="8" location="A2" yy="2022" quantity="150" status="NO" mm="5" mrp="7" name="Women's Multivate" id="4" dd="20"/>

</medicines>

**Transactions.XML**

<Transactions>

 <Transaction Cashier="RamBabu" Date="17/12/2022" Time="21:20:24" BillNo="1">

  <Customer CustomerName="tony" CustomerID="486"/>

  <Orders TotalValue="20 Rs">

   <Order ProductName="abc" Mrp="2" ProductID="123" Quantity="10"/>

   <Order ProductName="def" Mrp="3" ProductID="124" Quantity="15"/>

   <Order ProductName="ghi" Mrp="4" ProductID="125" Quantity="20"/>

  </Orders>

 </Transaction>

 <Transaction Cashier="RamBabu" Date="17/12/2022" Time="22:31:20" BillNo="2">

  <Customer CustomerName="vinay" CustomerID="1234"/>

  <Orders TotalValue="16 Rs">

   <Order ProductName="jkl" Mrp="5" ProductID="126" Quantity="9"/>

   <Order ProductName="mno" Mrp="6" ProductID="127" Quantity="10"/>

   <Order ProductName="pqr" Mrp="7" ProductID="128" Quantity="10"/>

   <Order ProductName="stu" Mrp="8" ProductID="129" Quantity="10"/>

  </Orders>

 </Transaction>

 <Transaction Cashier="RamBabu" Date="17/12/2022" Time="21:20:24" BillNo="3">

  <Customer CustomerName="tony" CustomerID="486"/>

  <Orders TotalValue="20 Rs">

   <Order ProductName="abc" Mrp="2" ProductID="123" Quantity="10"/>

   <Order ProductName="def" Mrp="3" ProductID="124" Quantity="15"/>

   <Order ProductName="ghi" Mrp="4" ProductID="125" Quantity="20"/>

  </Orders>

 </Transaction>

 <Transaction Cashier="RamBabu" Date="18/12/2022" Time="11:23:43" BillNo="4">

  <Customer CustomerName="vinay" CustomerID="123"/>

  <Orders TotalValue="18 Rs">

   <Order ProductName="dolo 650" Mrp="2" ProductID="0" Quantity="10"/>

  </Orders>

 </Transaction>

 <Transaction Date="19/12/2022" Time="14:22:24" BillNo="5">

  <Customer CustomerName="chandana" CustomerID="451"/>

  <Orders TotalValue="8 Rs">

   <Order ProductName="dolo 650" Mrp="2" ProductID="0" Quantity="10"/>

   <Order ProductName="Probiotics" Mrp="3" ProductID="1" Quantity="5"/>

   <Order ProductName="Vitamin C" Mrp="4" ProductID="2" Quantity="2"/>

  </Orders>

 </Transaction>

 <Transaction Date="19/12/2022" Time="15:51:11" BillNo="6">

  <Customer CustomerName="bole" CustomerID="427"/>

  <Orders TotalValue="14 Rs">

   <Order ProductName="dolo 650" Mrp="2" ProductID="0" Quantity="10"/>

  </Orders>

 </Transaction>

 <Transaction Date="20/12/2022" Time="15:26:43" BillNo="7">

  <Customer CustomerName="prahan" CustomerID="451"/>

  <Orders TotalValue="18 Rs">

   <Order ProductName="dolo 650" Mrp="2" ProductID="0" Quantity="10"/>

  </Orders>

 </Transaction>

 <Transaction Cashier="123" Date="20/12/2022" Time="21:10:2" BillNo="8">

  <Customer CustomerName="daddy" CustomerID="1234555"/>

  <Orders TotalValue="24">

   <Order ProductName="dolo 650" Mrp="2" ProductID="0" Quantity="12"/>

  </Orders>

 </Transaction>

</Transactions>

**Supplier.XML**

<suppliers>

 <supplier Name="Gowtham" company="gowtham pharma company" contact="123456789">

    <medicines>

        <medicine id="124" name="dolo 650" />

        <medicine id="234" name="Vitamin C" />

    </medicines>

 </supplier>

 <supplier Name="vinay" company="Vinay pharma company" contact="123456789">

    <medicines>

        <medicine id="682" name="med1" />

        <medicine id="451" name="med2" />

    </medicines>

     </supplier>

 <supplier Name="tony" company="Tony pharma company" contact="123456789">

    <medicines>

        <medicine id="741" name="med3" />

    </medicines>

     </supplier>

</suppliers>

**Login.XML**

<login>

 <Admin>

  <person username="vinay@srmap" password="vinay123"/>

 </Admin>

 <Employee>

  <person username="gowtham@srmap" password="gowtham123"/>

  <person username="tony@srmap" password="tony123"/>

  <person username="123" password="123"/>

 </Employee>

</login>