目　　录

一、任务要求

该系统为两种角色的用户提供服务，一种是餐厅管理员，一种是顾客。餐厅管理员根据账号、密码登录系统。顾客无需登录即可使用系统。

1、 顾客通过该餐厅在系统中提供的菜单为自己点餐。系统能够根据顾客的要求正确打出订单，订单内容包括订单编号、菜品名称、每个菜品的价格、份数、折扣等；订单分两种，一种是在店消费，在店消费要求包括餐桌号，是否有包厢费，另一种是外卖，外卖要求包括送餐时间，送餐地点，客户手机号，外卖服务费，成绩≥60；

2、 订单、用户信息保存在数据库中，其中，连接数据库所需信息（数据库服务器地址、用户名、密码、数据库名）存放在文件中，程序通过从文件中读取这些信息获得与数据库的连接。餐厅管理员可以根据订单编号或手机号查找、删除或修改某个订单，查询到的订单按照下单时间先后显示，成绩≥70；

3、 菜单信息保存在数据库中，能够实现对餐厅菜式和价格的管理，包括对菜品和对应价格的增加、修改、删除、查找，折扣的设置，设置后，顾客在点餐时看到的是新设置后的菜单，成绩≥80；

4、 系统可根据数据库中保存的订单记录对销售情况进行统计，根据餐厅管理员的输入日期统计某天的销售情况（包括一共接了多少单，销售额是多少，各个菜品的销售情况，外卖和在店销售的占比）并显示订单详情，成绩≥90；

二、业务流程图

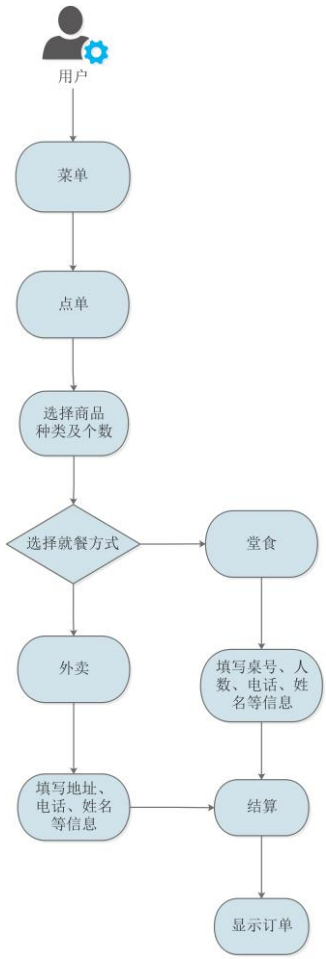


图 1 用户点餐流程图

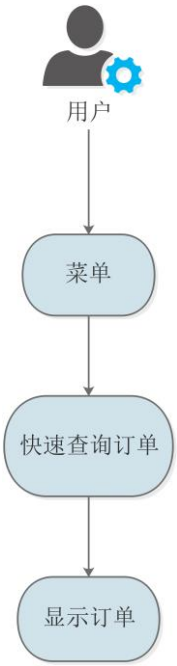


图 2 用户查询流程图

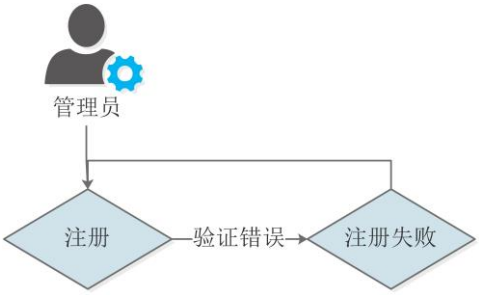


图 3 管理员注册流程图

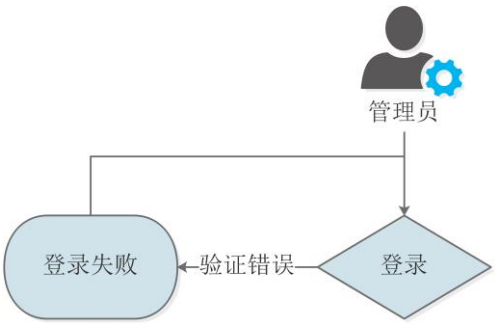


图 4 管理员登录流程图

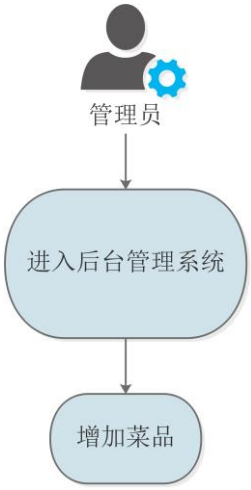


图 5 用户增添流程图

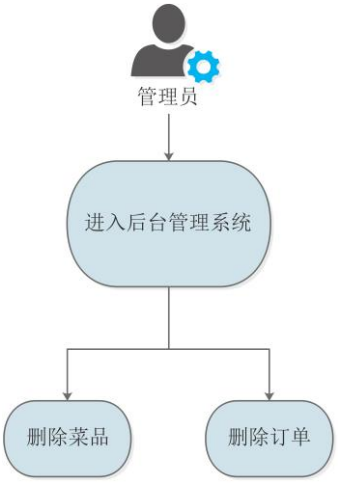


图 6 管理员删除流程图

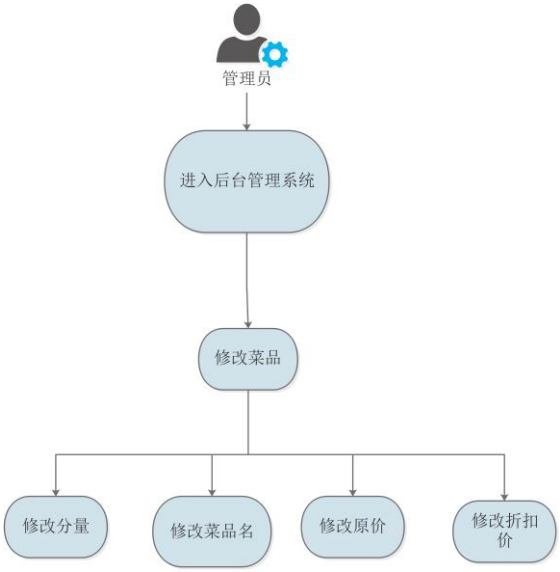


图 7 管理员修改流程图

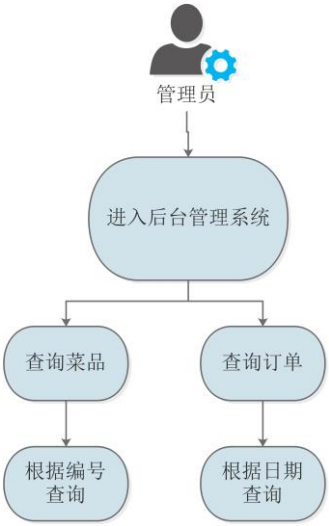


图 8 管理员查询流程图

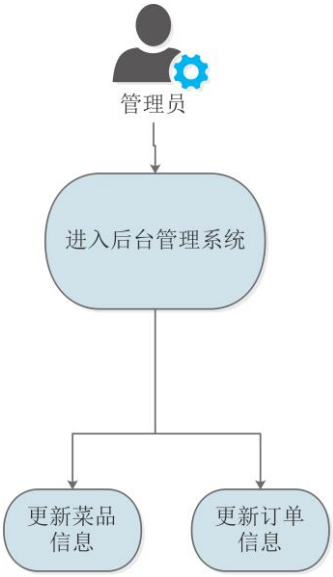


图 9 管理员更新流程图

三、系统功能结构图

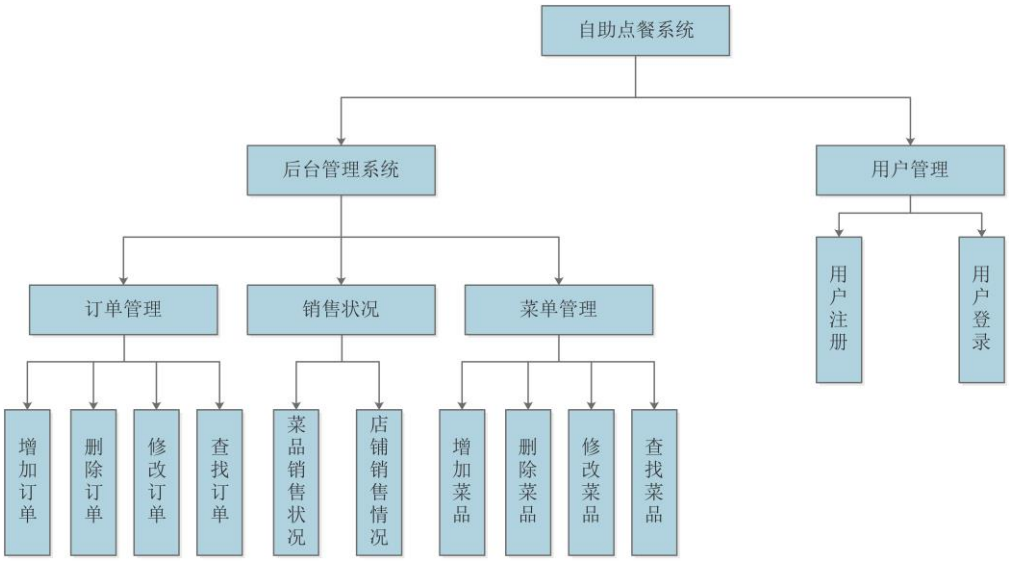


图 1 系统功能结构图

四、类的设计

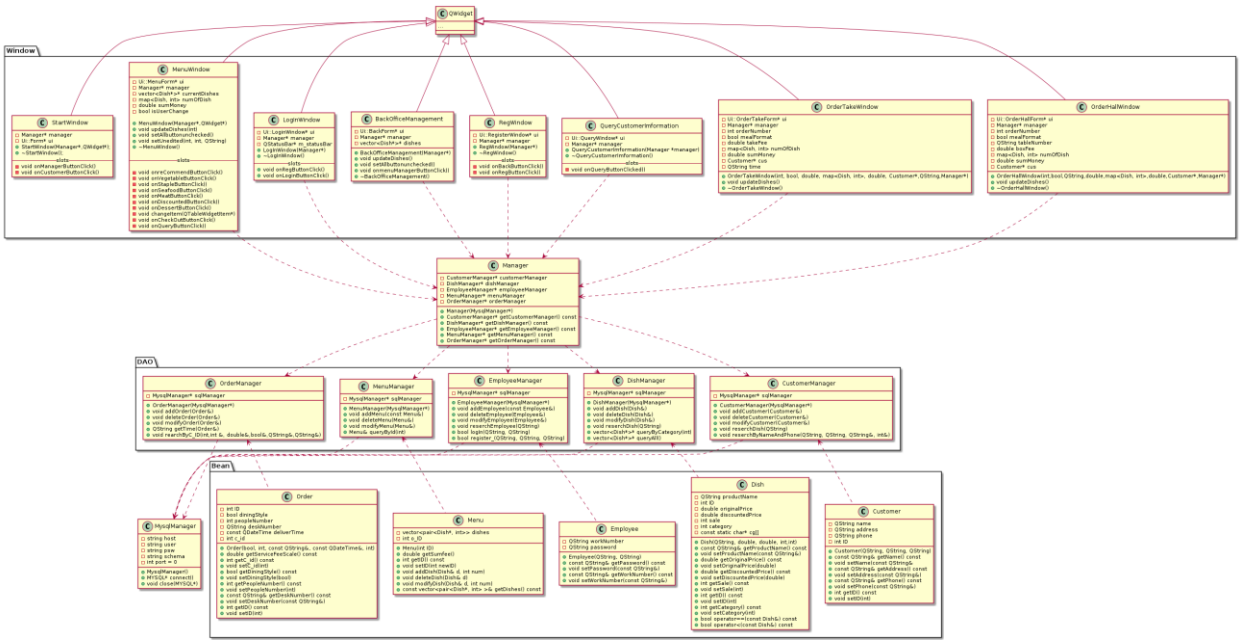


图 2 UML 图

五、数据库设计

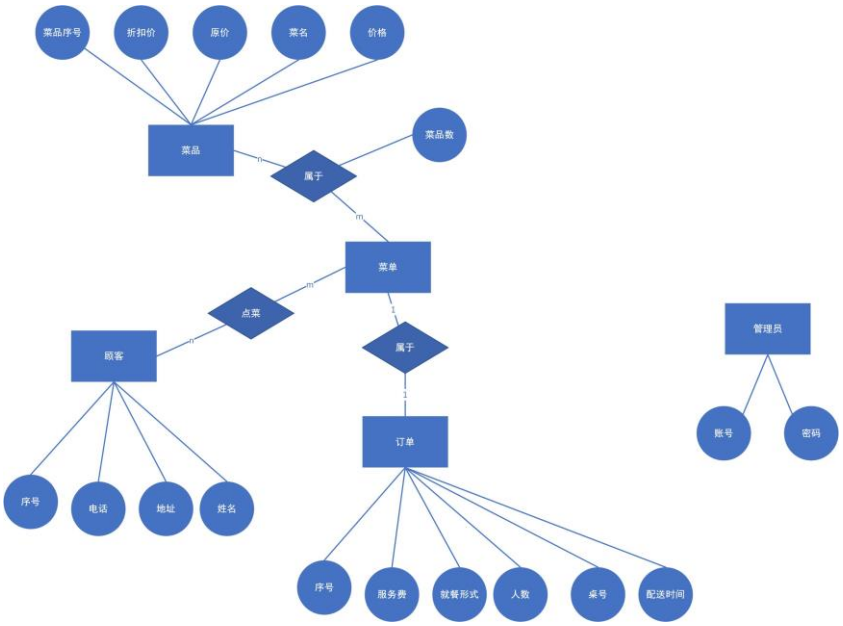


图 3 ER 图

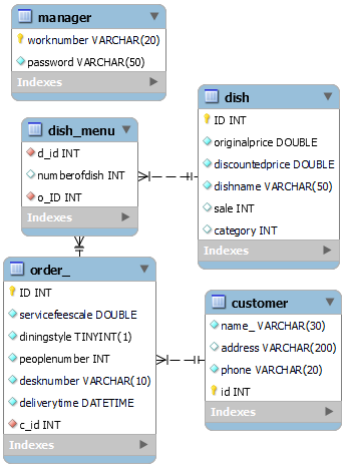


图 4 ER 图

```
CREATE TABLE `customer`  (
  `name_` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `address` varchar(200) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL,
  `phone` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `id` int NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE INDEX `un`(`name_` ASC, `phone` ASC) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 33 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `dish`  (
  `ID` int NOT NULL AUTO_INCREMENT,
  `originalprice` double NOT NULL,
```

```
  `discountedprice` double NOT NULL,
  `dishname` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `sale` int NULL DEFAULT 0,
  `category` int NULL DEFAULT NULL,
  PRIMARY KEY (`ID`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 3 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `dish_menu`  (
  `d_id` int NOT NULL,
  `numberofdish` int NULL DEFAULT 0,
  `o_ID` int NOT NULL,
  INDEX `d_id`(`d_id` ASC) USING BTREE,
  INDEX `o_ID`(`o_ID` ASC) USING BTREE,
  CONSTRAINT `dish_menu_ibfk_1` FOREIGN KEY (`d_id`) REFERENCES `dish` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT `dish_menu_ibfk_2` FOREIGN KEY (`o_ID`) REFERENCES `order_` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `manager`  (
  `worknumber` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `password` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  PRIMARY KEY (`worknumber`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

CREATE TABLE `order_`  (
  `ID` int NOT NULL AUTO_INCREMENT,
  `servicefeescale` double NOT NULL,
  `diningstyle` tinyint(1) NOT NULL,
  `peoplenumber` int NOT NULL,
  `desknumber` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `deliverytime` datetime NOT NULL,
  `c_id` int NOT NULL,
  PRIMARY KEY (`ID`) USING BTREE,
  INDEX `m_idx`(`c_id` ASC) USING BTREE,
  CONSTRAINT `a` FOREIGN KEY (`c_id`) REFERENCES `customer` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB AUTO_INCREMENT = 22 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
```

六、程序代码与说明

Customer.h

```
#ifndef CUSTOMER_H
#define CUSTOMER_H
#include <QString>

class Customer
{
    QString name;
    QString address;
    QString phone;
    int ID=0;
public:
    Customer(QString name, QString address, QString phone);
    const QString& getName() const;
    void setName(const QString& newName);
    const QString& getAddress() const;
    void setAddress(const QString& newAddress);
    const QString& getPhone() const;
    void setPhone(const QString& newPhone);
    int getID() const;
    void setID(int newID);
};
```

CustomerManager.h

```
#endif // CUSTOMER_H
#ifndef CUSTOMERMANAGER_H
#define CUSTOMERMANAGER_H
#include "MysqlManager.h"
#include "customer.h"
#include <QString>
#include "dish.h"
#include<iostream>
#include <map>
using std::map;

class CustomerManager
{
    MysqlManager* sqlManager;
public:
    CustomerManager(MysqlManager* sqlManager);
    //对于顾客进行增删改查
    void addCustomer(Customer& c);
    void deleteCustomer(Customer& c);
```

```
    void modifyCustomer(Customer& c);
    void reserchDish(QString str);
    void reserchByNameAndPhone(QString name, QString phone,QString &address, int& c_ID);
    Customer* reserchByID(int c_ID);
};

#endif // CUSTOMERMANAGER_H

#ifndef DISH_H
#define DISH_H
#include<QString>
```

Dish.h

```
class Dish {
    QString productName;
    int ID;
    double originalPrice;
    double discountedPrice;
    int sale;
    int category;
    const static char* cg[];
    //int leftNum;//剩余量
public:
    Dish(QString productName, double originalPrice, double discountedPrice, int sale,int category);

    const QString& getProductName() const;
    void setProductName(const QString& newProductName);
    double getOriginalPrice() const;
    void setOriginalPrice(double newOriginalPrice);
    double getDiscountedPrice() const;
    void setDiscountedPrice(double newDiscountedPrice);
    int getSale() const;
    void setSale(int newSale);
    int getID() const;
    void setID(int newID);
    int getCategory() const;
    void setCategory(int newCategory);
    bool operator==(const Dish& dish) const;
    bool operator<(const Dish& dish) const;
    //int getLeftNum() const;
    //void setLeftNum(int leftNum);
    static const char * getCg(int category);

};
```

Dishmanager.h

```
#endif // DISH_H

#ifndef DISHMANAGER_H
#define DISHMANAGER_H
#include <QString>
#include <iostream>
#include <vector>
#include"dish.h"
#include "MysqlManager.h"

using std::vector;

class DishManager {
    MysqlManager* sqlManager;
public:
    DishManager(MysqlManager* sqlManager);
    //增删改查
    void addDish(Dish& d);
    void deleteDish(int id);
    void modifyDishByID(Dish& d);
    void reserchDish(QString str);
    vector<Dish*>* queryByCategory(int cg);
    vector<Dish*>* queryAll();
};
```

Employee.h

```
#endif // DISHMANAGER_H

#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <QString>

class Employee {
    QString workNumber;
    QString password;
public:
    Employee(QString workNumber, QString password);

    const QString& getPassword() const;
    void setPassword(const QString& newPassword);
    const QString& getWorkNumber() const;
    void setWorkNumber(const QString& newWorkNumber);
};
```

Employeemanager.h

```
#endif // EMPLOYEE_H

#ifndef EMPLOYEEMANAGER_H
#define EMPLOYEEMANAGER_H
#include<QString>
#include "employee.h"
#include "MysqlManager.h"

class EmployeeManager {
    MysqlManager* sqlManager;
public:
    EmployeeManager(MysqlManager* sqlManager);
    //增删改查
    void addEmployee(const Employee& e);
    void deleteEmployee(Employee& e);
    void modifyEmployee(Employee& e);
    void reserchEmployee(QString str);
    //登录后台管理系统
    bool logIn(QString workNumber, QString passWord);
    //注册账号
    bool register_(QString workNumber, QString password, QString passwordAgain);
};

#endif // EMPLOYEEMANAGER_H
```

Manager.h

```
#pragma once
#include "customermanager.h"
#include "dishmanager.h"
#include "employeemanager.h"
#include "menumanager.h"
#include"ordermanager.h"
#include "MysqlManager.h"

class Manager
{
    CustomerManager* customerManager;
    DishManager* dishManager;
    EmployeeManager* employeeManager;
    MenuManager* menuManager;
    OrderManager* orderManager;
public:
    Manager(MysqlManager* mm);
```

```
    CustomerManager* getCustomerManager() const;
    DishManager* getDishManager() const;
    EmployeeManager* getEmployeeManager() const;
    MenuManager* getMenuManager() const;
    OrderManager* getOrderManager() const;
};
```

Menu.h

```
#pragma once
#include "customermanager.h"
#include "dishmanager.h"
#include "employeemanager.h"
#include "menumanager.h"
#include"ordermanager.h"
#include "MysqlManager.h"

class Manager
{
    CustomerManager* customerManager;
    DishManager* dishManager;
    EmployeeManager* employeeManager;
    MenuManager* menuManager;
    OrderManager* orderManager;
public:
    Manager(MysqlManager* mm);
    CustomerManager* getCustomerManager() const;
    DishManager* getDishManager() const;
    EmployeeManager* getEmployeeManager() const;
    MenuManager* getMenuManager() const;
    OrderManager* getOrderManager() const;
};
```

Menumanager.h

```
#ifndef MENUMANAGER_H
#define MENUMANAGER_H

#include <QString>
#include "menu.h"
#include "MysqlManager.h"

class MenuManager
{
    MysqlManager* sqlManager;
public:
```

```
    MenuManager(MysqlManager* sqlManager);
    void addMenu(const Menu& m);
    void deleteMenu(Menu& m);
    void modifyMenu(Menu& m);
    Menu& queryById(int o_id);

};

#endif // MENUMANAGER_H
```

Menuwindow.h

```
#pragma once

#include <qwidget.h>
#include "dishmanager.h"
#include "manager.h"
#include <iostream>
#include <map>
#include <QTableWidgetItem>

using std::map;

namespace Ui {
    class MenuForm;
}

class MenuWindow:public QWidget
{
    Q_OBJECT
private:
    Ui::MenuForm* ui;
    Manager* manager;
    vector<Dish*>* currentDishes=nullptr;
    map<Dish, int> numOfDish;
    double sumMoney = 0;
    bool isUserChange = true;

public:
    MenuWindow(Manager* manager,QWidget* parent = nullptr);
    void updateDishes(int category);
    void setAllbuttonunchecked();
    void setUnedited(int row, int currentrow, QString text);
    ~MenuWindow();
```

```
private slots:
    void onreCommendButtonClick();
    void onVegetableButtonClick();
    void onStapleButtonClick();
    void onSeafoodButtonClick();
    void onMeatButtonClick();
    void onDiscountedButtonClick();
    void onDessertButtonClick();
    void changeItem(QTableWidgetItem* item);
    void onCheckOutButtonClick();
    void onQueryButtonClick();
};
```

Mysqlmanager.h

```
#ifndef CONNECTMYSQL_H
#define CONNECTMYSQL_H
#include <mysql.h>
#include <string>

using std::string;

class MysqlManager
{
    string host;
    string user;
    string psw;
    string schema;
    int port = 0;
public:
    MysqlManager();
    MYSQL* connect();
    void close(MYSQL* mysql);

};

#endif // CONNECTMYSQL_H
```

Order.h

```
#ifndef ORDER_H
#define ORDER_H
#include"dish.h"
#include<vector>
using std::vector;
```

```
class Order
{
    int ID;
    bool diningStyle;//0为堂食，1为外卖
    int peopleNumber;
    QString deskNumber;
    QString deliverTime;
    int c_id;
public:
    Order(bool diningStyle, int peopleNumber, const QString& deskNumber, const QString& deliverTime,
int c_id);

    //根据就餐方式，判断服务费或是外卖费，再根据桌号判断就餐费
    double getServiceFeeScale() const;

    int getC_id() const;
    void setC_id(int newC_id);
    bool getDiningStyle() const;
    void setDiningStyle(bool newDiningStyle);
    int getPeopleNumber() const;
    void setPeopleNumber(int newPeopleNumber);
    const QString& getDeskNumber() const;
    void setDeskNumber(const QString& newDeskNumber);
    int getID() const;
    void setID(int newID);
    const QString getDeliverTime() const;
    void setDeliverTime(const QString deliverTime);

};

#endif // ORDER_H
```

Orderhallwindow.h

```
#pragma once

#include "qwidget.h"
#include "manager.h"
#include <iostream>
#include <map>

using std::map;

namespace Ui {
    class OrderHallForm;
```

```
}

class OrderHallWindow : public QWidget
{
    Q_OBJECT
private:
    Ui::OrderHallForm* ui;
    Manager* manager;
    int orderNumber;
    bool mealFormat;
    QString tableNumber;
    double boxFee;
    map<Dish, int> numOfDish;
    double sumMoney;
    Customer* cus;
public:
    OrderHallWindow(int orderNumber,bool mealFormat,QString tableNumber,double boxFee,map<Dish, int>
numOfDish,double sumMoney,Customer*cus,Manager* manager);

    void updateDishes();
    ~OrderHallWindow();
};

Ordermanager.h
#ifndef ORDERMANAGER_H
#define ORDERMANAGER_H
#include "MysqlManager.h"
#include "order.h"

class OrderManager
{
    MysqlManager* sqlManager;
public:
    OrderManager(MysqlManager* sqlManager);

    void addOrder(Order& o);
    void deleteOrder(int o_ID);
    void modifyOrder(Order& o);
    QString getTime(Order& o);
    void rearchByC_ID(int c_ID,int &o_ID, double& serviceFeeScale,bool &diningStyle,QString
&deskNumber,QString &time);
    vector<Order*>* rearchAll(vector<int>* o_ID, vector<int>* c_ID);
    vector<Order*>* rearchByTime(QString time, vector<int>* o_ID, vector<int>* c_ID);
    vector<int>* rearchByC_ID();
```

```
};

#endif // ORDERMANAGER_H
```

Ordertakewindow.h

```
#pragma once

#include "qwidget.h"
#include "manager.h"
#include <iostream>
#include <map>
```

OrderTakeWindow.h

```
using std::map;

namespace Ui {
    class OrderTakeForm;
}

class OrderTakeWindow : public QWidget
{
    Q_OBJECT
private:
    Ui::OrderTakeForm* ui;
    Manager* manager;
    int orderNumber;
    bool mealFormat;
    double takeFee;
    map<Dish, int> numOfDish;
    double sumMoney;
    Customer* cus;
    QString time;
public:
    OrderTakeWindow(int orderNumber, bool mealFormat, double takeFee, map<Dish, int> numOfDish, double
sumMoney, Customer* cus,QString time,Manager*manager);

    void updateDishes();
    ~OrderTakeWindow();
};

#pragma once
#include "qwidget.h"
#include "manager.h"
```

```
namespace Ui {
    class QueryWindow;
};

class QueryCustomerImformation:public QWidget
{
    Q_OBJECT
;private:
    Ui::QueryWindow* ui;
    Manager* manager;
public:
    QueryCustomerImformation(Manager *manager);
    ~QueryCustomerImformation();
private slots:
    void onQueryButtonClicked();
};
```

Startwindow.h

```
#ifndef STARTWINDOW_H
#define STARTWINDOW_H

#include <QWidget>
#include "manager.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class Form;
}
QT_END_NAMESPACE

class StartWindow:public QWidget
{
    Q_OBJECT
private:
    Manager* manager;
    Ui::Form* ui;
public:
    StartWindow(Manager* manager,QWidget* parent = nullptr);
    ~StartWindow();

private slots:
    void onManagerButtonClick();
    void onCustomerButtonClick();
};
```

BackOfficeManagement.h

```
#endif // STARTWINDOW_H

#pragma once
#include "manager.h"
#include <qwidget.h>
#include<QTableWidgetItem>

namespace Ui {
    class BackForm;
}

class BackOfficeManagement:public QWidget
{
    Q_OBJECT
private:
    int orderNum=0;
    double sumMoney=0;
    int divisionNum=0;
    Ui::BackForm* ui;
    Manager* manager;
    vector<Dish*>* dishes=nullptr;
    vector<Order*>* orders;
    vector<Customer*>* cus;
    vector<int>* o_id;
    vector<int>* c_id;
    bool isUserChange =true;
public:
    BackOfficeManagement(Manager* manager);
    void updateDishes();
    void setAllbuttonunchecked();
    void updateOrder();
    ~BackOfficeManagement();
private slots:
    void onMenuManagerButtonClick();
    void onOrderManagerButtonClick();
    void addItem();
    void deleteItem();
    void changeItem(QTableWidgetItem* item);
};

#pragma once
```

ComboItemDelegate.h

```
#include <QItemDelegate>
#include <QStringList>
#include <QStyledItemDelegate>
#include <QValidator>
#include <QWidget>

class ComboItemDelegate : public QStyledItemDelegate {
    Q_OBJECT
        QStringList items;

public:
    ComboItemDelegate(QStringList items, QObject* parent = 0);

    QWidget* createEditor(QWidget* parent, const QStyleOptionViewItem& option,
        const QModelIndex& index) const;

    void setEditorData(QWidget* editor, const QModelIndex& index) const;
    void setModelData(QWidget* editor, QAbstractItemModel* model,
        const QModelIndex& index) const;

    void updateEditorGeometry(QWidget* editor,
        const QStyleOptionViewItem& option,
        const QModelIndex& index) const;
};

#pragma once
#include <qwidget.h>
#include"manager.h"
#include<QStatusBar>

LogInWindow.h
namespace Ui {
     class LoginWindow;
}

class LogInWindow:public QWidget
{
    Q_OBJECT
private:
    Ui::LoginWindow* ui;
    Manager* manager;
    QStatusBar* m_statusBar;
public:
    LogInWindow(Manager* manager);
```

```
    ~LogInWindow();
private slots:
    void onRegButtonClick();
    void onLoginButtonClick();
};
```

RegWindow.h

```
#pragma once
#include <qwidget.h>
#include"manager.h"

namespace Ui {
    class RegisterWindow;
}

class RegWindow:public QWidget
{
    Q_OBJECT
private:
    Ui::RegisterWindow* ui;
    Manager* manager;
public:
    RegWindow(Manager* manager);
    ~RegWindow();
private slots:
    void onBackButtonClick();
    void onRegButtonClick();
};
```

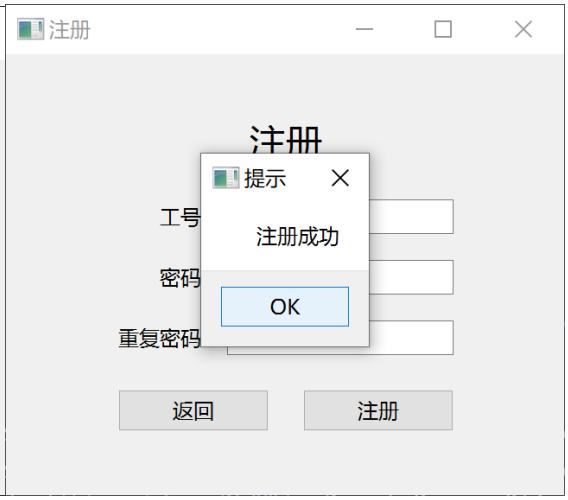
七、运行结果与分析


石榴裙自主点餐系统
石榴裙自助点餐系统
用户
管理员

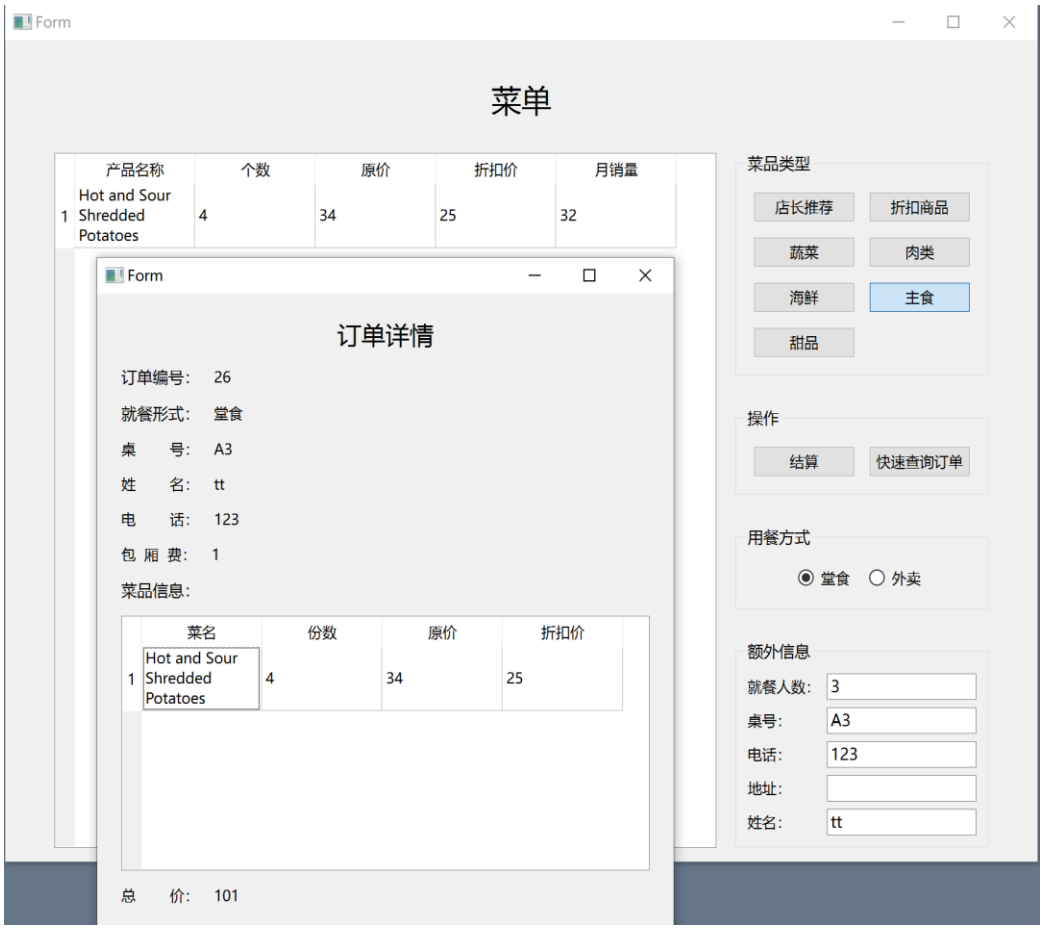

Form
后台管理
产品编号
产品名称
原价
折扣价
种类
销量
1 11 麻婆豆腐 25 23 折扣商品 7
2 12 肉末茄子 29 29 肉类 11
3 13 冰淇淋面包 25 25 甜品 6
4 15 米饭 2 2 主食 7
菜品类型
菜单管理
订单管理
操作
增加
删除


Form
订单详情
订单编号: 23
就餐形式: 外卖
姓 名: slq
电 话: 123
地 址: zjlgdx
配 送 费: 4
预计送达时间: 2022-01-12 01:44:17
菜品信息:
菜名
份数
原价
折扣价
总 价: 0


登录
后台管理系统
工号
111
密码
注册
登录


注册
注册
工号
密码
重复密码
返回
注册
提示
注册成功
OK


Form
后台管理
请选择日期 2022/1/12
订单编号 附加费用 就餐形式 桌号 姓名 地址 预计送达时间 手机号
1 27 0.03 外卖 B5 qqq 2022-01... 08:23:58 12321
2 28 0.03 外卖 B3 syc 2022-01... 08:32:05 13321
3 29 0.03 外卖 B2 slq 2022-01... 08:45:21 12332
4 30 0.03 堂食 syc 浙江理工大学 2022-01... 08:46:04 35216
菜品类型
菜单管理
订单管理
操作
增加
删除
销售数据
接单数: 4
销售额: 644
外卖/在店: 1:3


Form
菜单
产品名称 个数 原价 折扣价 月销量
1 Hot and Sour Shredded Potatoes 4 34 25 32
菜品类型
店长推荐
折扣商品
蔬菜
肉类
海鲜
主食
甜品
操作
结算
快速查询订单
用餐方式
堂食
外卖
额外信息
就餐人数: 3
桌号: A3
电话: 123
地址:
姓名: tt
Form
订单详情
订单编号: 26
就餐形式: 堂食
桌 号: A3
姓 名: tt
电 话: 123
包 厢 费: 1
菜品信息:
菜名 份数 原价 折扣价
1 Hot and Sour Shredded Potatoes 4 34 25
总 价: 101

八、心得与体会

1、 运用了 QT

学会了使用 QT 来实现程序的可视化，便于与用户的交互，并实现了程序的简易性与可重复利用性。并且不仅是尝试去运用 QT 的 GUI，还尝试了用代码实现对于窗口或是控件的书写。这相较于传统的命令行，会更加地有趣，并极大的便利了开发者。

2、 连接数据库

一开始尝试用 QT 与数据库直连，但由于安装版本过高，网上教程较少，最终没有成功，但在中途多次尝试，虽说没有成功，但也学到了许多知识。最后运用 c++的 mysql 库，实现了功能。

3、 数据库的学习

在本学期由于学习了数据库，因此，这次课设对于我来说，也是一次对于数据库学习的检验，更增强了我对于数据库知识的巩固。

4、 学习了项目相关的画图软件的应用

在本次实验中，第一次接触了类图、流程图与项目管理图，更加体会到一项工程的产生多么来之不易，也对于 c++这类编程语言，产生了更加深刻的理解。

5、 使用了表达式

运用正则表达式，对于数据的输入进行限制，以便于用户的交互。

6、 异常处理

通过课上所学习的异常处理，来运用于程序当中，实现异常处理机制，从而解决多数无法用异常处理能够迎刃而解的问题。