目录

1课题分析..........................................................3

 1.1课题背景......................................................3

 1.2研究目的及意义................................................3

 1.3本文所做的工作................................................4

2开发技术、开发工具及技术路线......................................5

 2.1开发技术及开发工具............................................5

  2.1.1开发技术...............................................5

  2.2.2开发工具...............................................6

 2.2技术路线......................................................6

3功能分析与设计....................................................7

 3.1需求分析......................................................7

  3.1.1用户调研...............................................7

  3.1.2初步设计...............................................7

 3.2系统设计......................................................7

 3.3数据库概念结构设计............................................7

3.4数据库逻辑结构设计............................................8

3.4.4 E-R图向关系模型的转换.................................8

3.5数据库物理结构设计............................................9

3.6模块划分......................................................9

4实现.............................................................10

4.1前端界面实现.................................................10

4.1.1管理员界面实现.........................................11

4.2后端代码实现.................................................13

4.2.1 C#与SQL Server的连接和交互............................13

4.2.2 SQL Server建立数据库..................................20

5系统测试与应用...................................................26

 5.1系统测试运行环境.............................................26

 5.2数据库测试与运行.............................................26

  5.2.1点菜系统的点菜过程........................................26

5.2.1.1用户登录（已有默认账号）.............................26

5.2.1.2根据提示输入信息，进入点菜界面.......................26

5.2.1.3 进行餐桌预约.......................................27

5.2.1.4 预定成功后根据提示点菜.............................28

5.2.1.5 根据提示进行下单付款...............................29

5.2.2各列表的查看与修改........................................30

5.2.2.1顾客列表查看........................................30

5.2.2.2帐单列表查看........................................30

5.2.2.3餐桌列表查看........................................30

5.2.2.4增加或修改顾客信息..................................30

5.2.2.5员工列表查看........................................31

5.2.2.6员工列表的修改......................................32

5.2.2.7菜品列表查看........................................33

5.2.2.8退出系统............................................34

6总结与展望.......................................................34

6.1总结.........................................................34

6.2不足.........................................................35

 6.3展望.........................................................35

餐馆点菜管理系统

摘要：本课程设计报告详细地描述了本小组开发出的一套“餐馆点菜管理系统”软件，从课题背景入手，分析了开发的目的和意义；接着阐明了本设计的开发技术路线以及功能分析与设计，包括需求分析、系统设计、概念结构设计等；最后依据开发路线和设计实现了互助平台，并对应用程序和数据库进行了测试；

关键词：餐馆管理；数据管理；数据上传；C#；SQL server

引言：随着网络技术的发展以及物质生活水平的提高，现代的餐饮行业日益繁荣，所需要处理的订单信息量也随之增大。为了使的餐馆能够用更少的成本达到同样的效果，如果有那么一款软件能够代替一部分认为点单，那么我们将节省更多的人力资源。

**1 课题分析**

1.1 课题背景

在日益繁华的生活中，人们的消费水平不断地提高，越来越多的人会选择去外面就餐，这也就导致餐馆的类似于点菜与该桌是否有人预定等情况需要进行相应的记录，但是认为的记录到本子上在查看时也有诸多的不便，而且餐馆的人员数量的增加导致餐馆的人员管理更加的困难，所以以往的人员记录方式不够迅速便捷。所以餐馆的管理人员需要一个管理系统来帮助他更好的管理餐馆，从而在生产过程中减少不必要的错误以及增加餐馆的利润。也许曾经的点餐方式在人流量较少的时候可以应付的过来，但是当人流量增多到一定的程度的时候，曾经的点餐的方式就显得比较笨重，点餐员可能会当顾客达到餐馆后的第一时间无法在旁边点餐，从而导致了顾客的流失，老式的点餐方式需要打单子的时间，而且增减菜的效率低，所以为了让餐馆更好的留住顾客使得餐馆的效益最大化，于是我们想到做出一个餐馆点菜系统帮助餐馆进行人员的管理以及加快点菜的效率。

1.2 研究目的及意义

本课程设计的目的在于综合运用所学程序设计语言以及数据库原理的知识，开发以上说的餐馆点菜系统，进行测试并运行，以确保平台的可使用性。在开发的过程中复习所学的知识，并在实践过程中积累项目开发经验。

1.3 本文所做的工作

本文数据库系统的相关理论用于实践，在需求分析以及系统开发的过程中，具体所做的工作有：

1. 对于餐馆点菜管理系统的基本功能做了简单的分析了解，和实现系统的技术路线。
2. 在该系统的总体设计阶段，给出了系统实现的功能模块和基本架构。

（3）在详细设计和具体实施阶段完成了对于互助平台的功能模块设计、功能设 计、数据库设计，并完成实现了具体的功能

（4）在原有的基础上对餐馆点菜管理系统进行错误的修改与功能的优化，提高 管理人员使用的便捷性。

**2 开发技术、开发工具及技术路线**

2.1 开发技术及开发工具

2.1.1 开发技术

本平台是基于C#开发的，后台数据库采用Microsoft SQL server。

（1）C#

C#（C sharp）是[Microsoft公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%AC%E5%8F%B8/732128" \t "_blank)发布的一种面向对象的、运行于[.NET Framework](https://baike.baidu.com/item/.NET%20Framework)和.NET Core（完全开源，跨平台）之上的高级程序设计语言。它是Microsoft公司.NET windows网络框架的主角。其语法风格源自C/C++家族：继承了C的过程化编程语言和C++面向对象思想的强大，融合了Visual Basic的高效，具有易上手、功能强大的特性，一经推出便深受世界各地程序员的好评和喜爱。

C#是一种由事件驱动、可视化的编程语言，结合了[Visual B](https://baike.baidu.com/item/VB)asic简单的可视化操作，以其强大的操作能力、优雅的语法风格、创新的语言特性和便捷的面向[组件](https://baike.baidu.com/item/%E7%BB%84%E4%BB%B6" \t "_blank)编程的支持成为[.NET](https://baike.baidu.com/item/.NET)开发的首选语言。它使得程序员可以快速地编写基于Microsoft .NET平台的应用程序，Microsoft .NET提供了一系列的[工具](https://baike.baidu.com/item/%E5%B7%A5%E5%85%B7)和服务来最大程度地开发利用计算与通讯领域。

C#使得C++程序员可以高效的开发程序，且因可调用由 C/C++ 编写的本机原生函数，而绝不损失C/C++原有的强大的功能。因为这种继承关系，C#与C/C++具有极大的相似性，熟悉类似语言的开发者可以很快的转向C#。

（2）SQL server

SQL是英文Structured Query Language的缩写，意思为[结构化查询语言](https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/10450182)。SQL语言的主要功能就是同各种数据库建立联系，进行沟通。按照ANSI（[美国国家标准协会](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E5%8D%8F%E4%BC%9A)）的规定，SQL被作为[关系型数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/696511" \t "_blank)的标准语言。SQL Server是由Microsoft开发和推广的[关系数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/11032386" \t "_blank)（RDBMS）。

SQL Server是由Microsoft开发和推广的关系型数据库，它最初是由Microsoft、Sybase和Ashton-Tate三家公司共同开发的，并于1988年推出了第一个OS/2版本。目前最新版本是2017年份推出的SQL Server 2017，也就是本课程设计使用的版本。

所谓数据库，就是存储数据的地方，需要长期存储在计算机中。数据库中的数据按照一定的数据模型组织、存储和描述，具有较小的冗余度、较高的数据独立性和易扩展性、可以为不同的用户所共享。

在诸多类型的数据库中，关系数据库由于其建立在严格的数学概念基础之上，概念单一，结构简单、清晰、用户易懂易用，存取路径等物理操作对用户透明等特点，成为当今的最主流数据库技术。

关系数据库是由数据表和表之间的关联组成的。这其中的数据表通常是一个由列和行组成的二维表。每一行乘坐一个元组，每一列称为属性，属性是数据库表中存储对象的共有属性。

对数据库的操作可通过数据库管理系统（Database Management System，简称DBMS）实现，它是数据库的核心组成部分，用户对数据库的一切操作，比如定义、查询、更新、插入、添加触发器等控制都是通过DBMS来进行操作的。Microsoft公司也为SQL Server开发了一套专属的DBMS——SQL Server Management Studio（SSMS），本课程设计使用的版本是17.9.1。

2.2.2 开发工具

本系统在Windows 10操作系统上开发，采用Microsoft Visual Studio 2017集成开发环境进行开发，数据库管理系统采用Microsoft SQL Server Management Studio 17.9.1。

（1）Microsoft Visual Studio

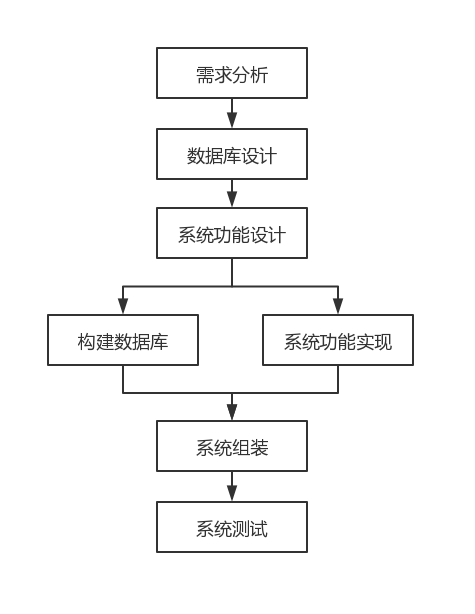
Microsoft Visual Studio，简称VS。VS是美国[微软公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%AC%E5%8F%B8" \t "_blank)的开发工具包系列产品。VS是一个基本完整的开发工具集，它包括了整个[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F" \t "_blank)中所需要的大部分工具，如UML工具、代码管控工具、[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83" \t "_blank)（IDE）等等。所写的[目标代码](https://baike.baidu.com/item/%E7%9B%AE%E6%A0%87%E4%BB%A3%E7%A0%81/9407934" \t "_blank)适用于微软支持的所有平台，包括[Microsoft Windows](https://baike.baidu.com/item/Microsoft%20Windows)、[.NET Framework](https://baike.baidu.com/item/.NET%20Framework" \t "_blank)等。[Visual Studio](https://baike.baidu.com/item/Visual%20Studio)是目前最流行的[Windows](https://baike.baidu.com/item/Windows" \t "_blank)平台应用程序的[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83" \t "_blank)。最新版本为 Visual Studio 2019版本，基于.NET Framework 4.7。本课程设计使用2017版本。

（2）Microsoft SQL Server Management Studio

对数据库的操作可通过数据库管理系统（Database Management System，简称DBMS）实现，它是数据库的核心组成部分，用户对数据库的一切操作，比如定义、查询、更新、插入、添加触发器等控制都是通过DBMS来进行操作的。Microsoft公司也为SQL Server开发了一套专属的DBMS——SQL Server Management Studio（简称SSMS），本课程设计使用的版本是17.9.1。

DBMS可以对[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)进行统一的管理和[控制](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6/948689)，以保证[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)的安全性和完整性。用户通过[DBMS](https://baike.baidu.com/item/DBMS)访问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)中的数据，[数据库管理员](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E5%91%98/1216449)也通过DBMS进行数据库的维护工作。它可使多个[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F/5985445" \t "_blank)和用户用不同的方法在同时或不同时刻去建立，修改和询问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)。大部分[DBMS](https://baike.baidu.com/item/DBMS)提供[数据定义语言](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%9A%E4%B9%89%E8%AF%AD%E8%A8%80/104009" \t "_blank)[DDL](https://baike.baidu.com/item/DDL/21997)（Data Definition Language）和[数据操作语言](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%93%8D%E4%BD%9C%E8%AF%AD%E8%A8%80/10826383" \t "_blank)[DML](https://baike.baidu.com/item/DML)（Data Manipulation Language），供用户定义[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)的模式结构与权限约束，实现对数据的追加、删除等操作。

数据库管理系统是数据库系统的核心，是管理数据库的软件。数据库管理系统就是实现把用户意义下抽象的逻辑数据处理，转换成为计算机中具体的物理数据处理的软件。有了数据库管理系统，用户就可以在抽象意义下处理数据，而不必顾及这些数据在计算机中的布局和物理位置。



2.2 技术路线

本系统在着重分析当前餐饮行业在点菜平台、人员管理平台缺点和不足的基础上，详细设计了针对餐馆点菜管理的功能，从而加快点菜和人员管理的效率，减少上错菜的可能性等。利用数据库平台、C#面向对象的程序设计等技术设计实现了餐馆点菜管理系统。

本平台的开发技术路线如图2-1所示：

**3 功能分析与设计**

3.1 需求分析

**3.1.1 用户调研**

本系统中的用户角色为来餐馆吃饭的顾客和平台管理人员，为了能够方便用户操作，在开发前需要针对实际情况进行详细分析，对各个模块间的跳转进行规划，让系统的功能和用户操作尽可能的符合实际的使用情况。

**3.1.2 初步设计**

根据系统的功能需求分析，系统的主要思路设计如下，这也是我们后续工作设计界面的依据：

（1）管理人员和点餐员工进行登录，登录的账号是已经创建好了的。

（2）点餐员工输入顾客手机号，若为新顾客则自动添加，为老顾客则直接进入点菜窗口。管理人员帮顾客预定餐桌，若餐桌已占用则显示已占用，若空闲则显示预定成功。之后就可以进入菜品列表供顾客选择菜品及数量。之后显示总金额然后按照提示下单付款。

（3）管理人员可以浏览顾客信息，餐品信息以及餐馆人员信息，并且可以对其进行修改。

3.2 系统设计

根据对需求分析阶段所得到的信息进行分析，将系统流程细化到如图3-1所示：

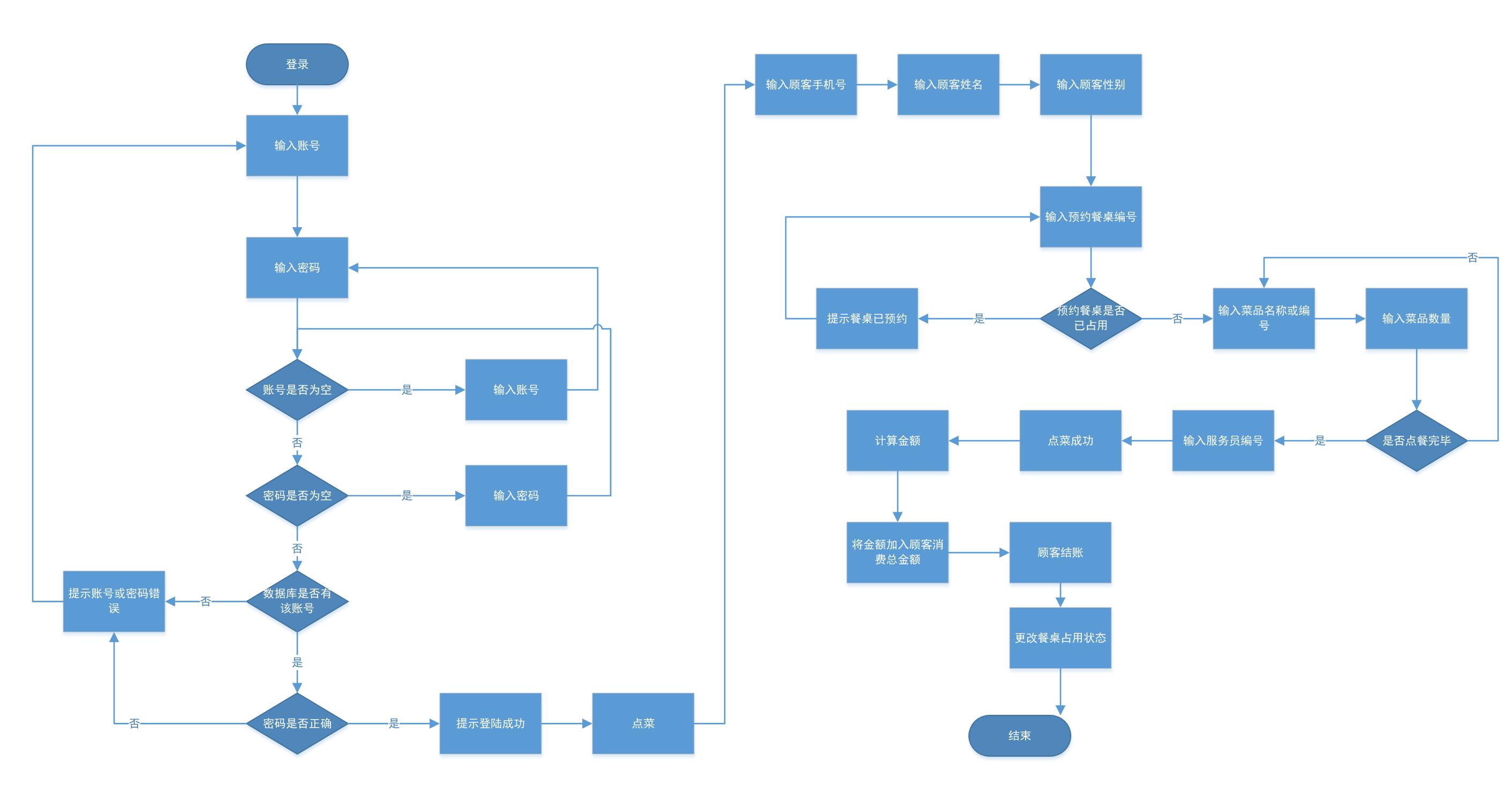


图3-1

3.3 数据库概念结构设计

在需求分析阶段所得到的应用需求应该首先抽象为信息世界的结构，然后才能更好、更准确地用某一数据库管理系统实现这些需求。概念模型是各种数据模型的共同基础。描述概念模型的有利工具E-R模型。

根据需求分析，我们建立了如下的E-R模型，如图3-3所示：

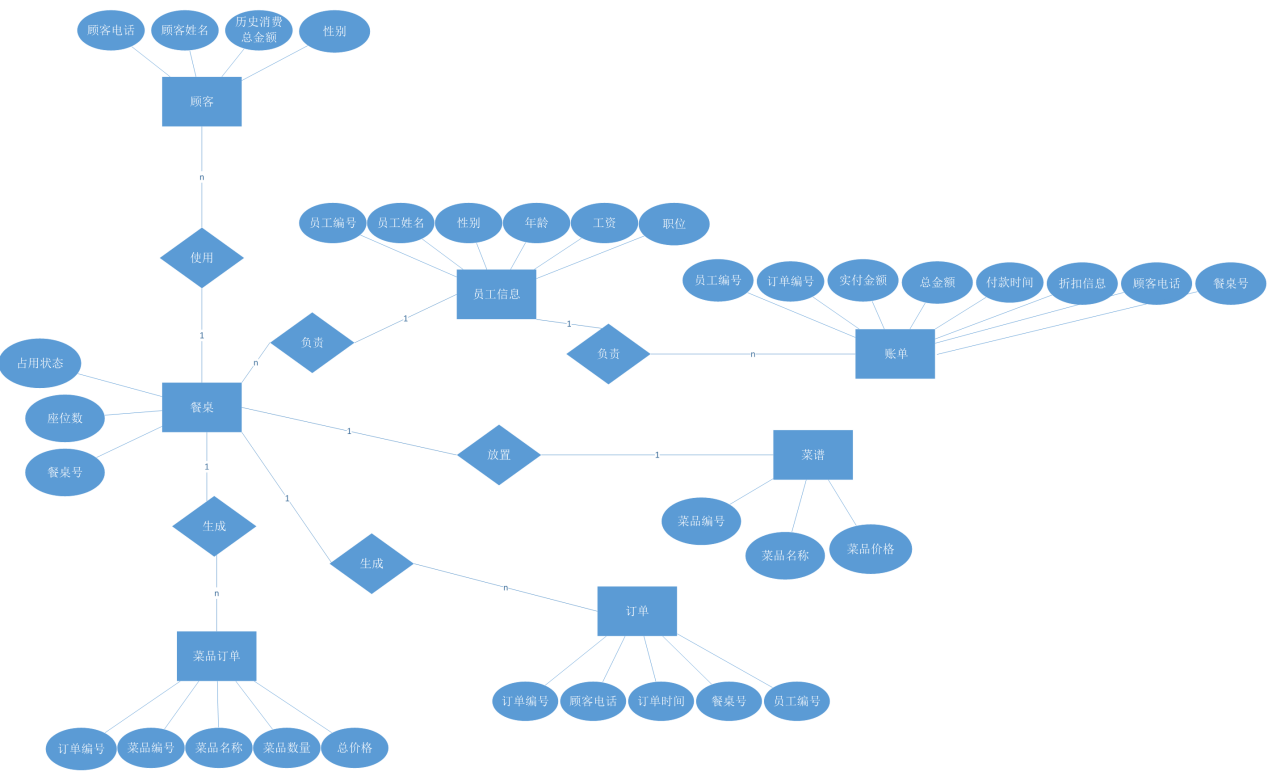


图3-3

3.4 数据库逻辑结构设计

数据库逻辑结构的设计就是把概念结构设计阶段设计好的基本E-R图转换为与选用DBMS所支持的数据模型相符合的逻辑结构。本课程设计采用的是关系型数据库，因此需要完成E-R图向关系模型的转换。

**3.4.1 E-R图向关系模型的转换**

根据图3-3所示的E-R模型图转换为关系模型，关系的码用下划线标出：

（1）账单{订单编号、员工编号、实付金额、总金额、付款时间、折扣信息、顾客电话、餐桌号}

（2）员工信息表 {员工编号、员工姓名、性别、年龄、工资、职位}

（3）菜谱表 {菜品编号、菜品名称、菜品价格}

（4）订单表 {订单编号、顾客电话、订单时间、餐桌号、员工编号}

（5）菜品订单表 {订单编号、菜品编号、菜品名称、菜品数量、总价格}

（6）餐桌表 {餐桌号、占用状态、座位数}

（7）顾客表 {顾客电话、顾客姓名、性别、历史消费总金额}

（8）用户信息表 {用户账号、用户密码、用户姓名}

3.5 数据库物理结构设计

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于选定的数据库管理系统。为一个给定的逻辑数据模型选取一个最适合应用要求的物理结构的过程，就是数据库的物理设计。

本平台采用的数据库平台是SQL Server，数据库以文件形式存储在计算机磁盘中，其中主要的文件有主数据文件，次数据文件和事物日志文件：

（1）主数据文件

主数据文件是数据库的起点，其中包含了数据库的初始信息，并记录数据库还拥有哪些文件。每个数据库有且只能有一个主数据文件。主数据文件是数据库必须的文件，主数据文件的扩展名是.mdf；

（2）次数据文件

除了主数据文件以外的所有其他数据文件都是次数据库文件。次数据文件不是数据库必需的文件。次数据文件的扩展名是.ndf；

（3）事务日记文件

每个数据库至少拥有一个自己的日记文件（也可以拥有多个日记文件）。日记文件的大小最少是1MB，默认扩展名是.ldf,用来记录数据库的事务以及每个数据对数据库所做的修改。

在SQL Server中，页是数据存储的基本单位，为数据库中的数据文件分配的磁盘空间可以从逻辑上划分带有连续编号的（编号从0开始）磁盘I/O操作在页级执行，SQL Server读取或写入的是所有的数据页。

3.6 模块划分

按照上述分析将系统最终划分为如下6个模块：

1. 管理人员/登录模块：

管理人员/点餐员填写已有账号、密码来，进入系统界面；

1. 员工管理模块：

管理人员可以对于员工的信息进行查看并修改：

1. 菜谱管理模块：

管理人员可以查看并修改菜品的信息；

1. 餐桌管理模块：

管理员与点餐员可以查看餐桌的占用情况；

1. 顾客管理模块：

管理员可以查看顾客的信息；

1. 账单查询模块：

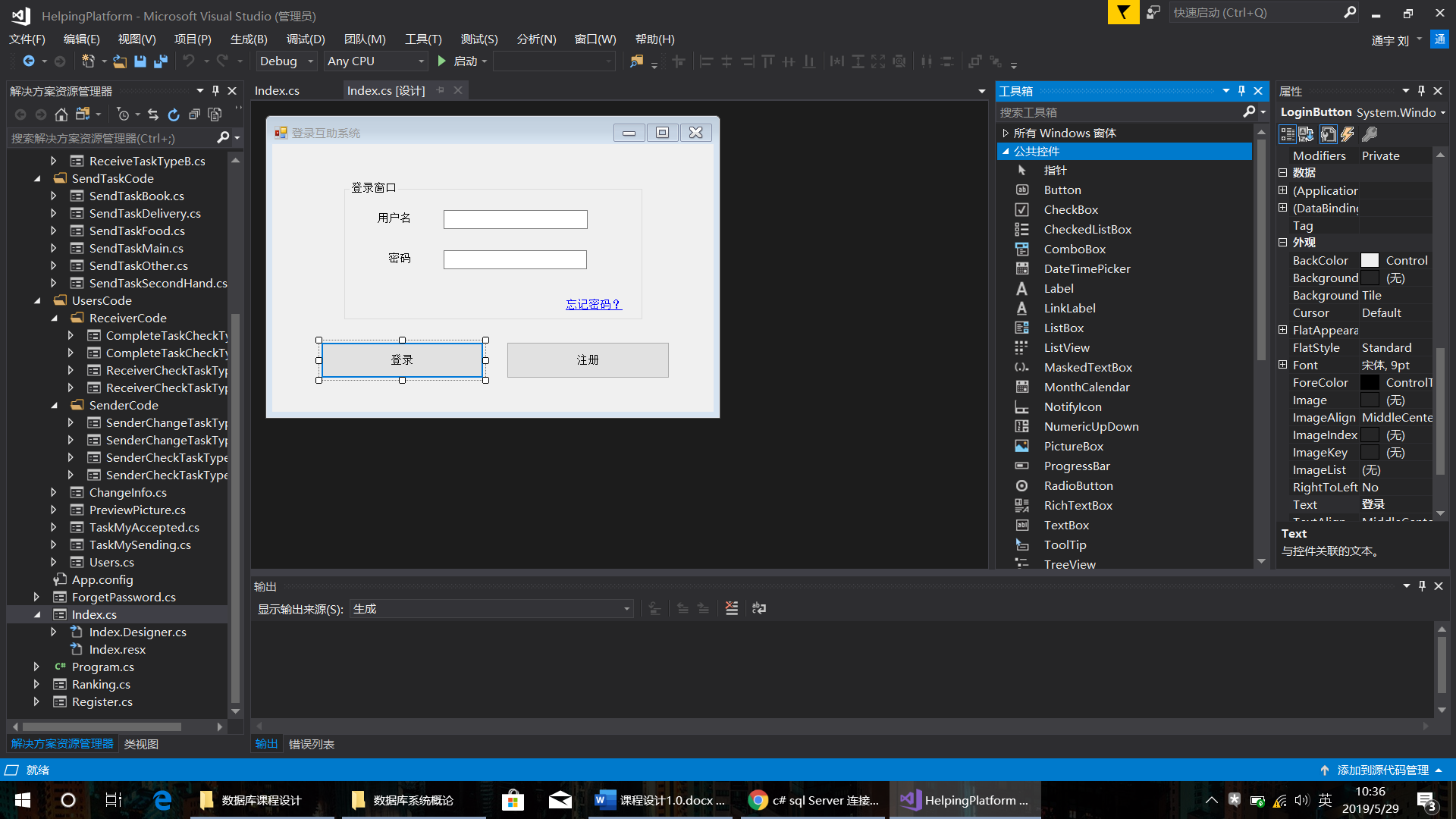
点餐员可以查看账单，可以按照顾客的需要对增减账单上的菜品与数量。

**4 实现**

4.1 前端界面实现

本平台前端界面开发主要借助Visual Studio 2017集成开发环境中针对C#窗体应用程序的设计器来完成前端界面的设计与开发工作。开发界面如图4-1所示。

利用这个设计器，可以所见即所得地、便捷地完成窗体的设计，只需要将所需的空间拖拽至窗体，再设置属性即可，设计器将自动地生成后缀名为.Designer.cs的文件，记录窗体以及控件属性信息，在编译时会自动构建窗体，不需要程序员手动编写代码来进行窗体设计，这为我们的开发工作提供了极大的方便。



**4.1.1 管理员界面实现**

管理员与点餐员工界面如图4-2-1至图4-2-6所示。

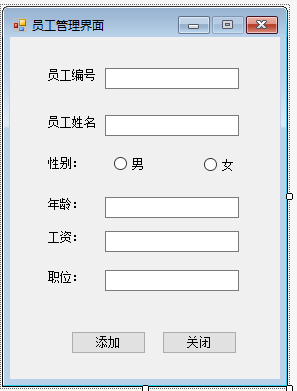


图4-2-1 登录界面 图4-2-2 员工管理界面



图4-2-3 饭店管理系统主界面



图4-2-4 顾客信息界面



图4-2-5 餐桌状态界面

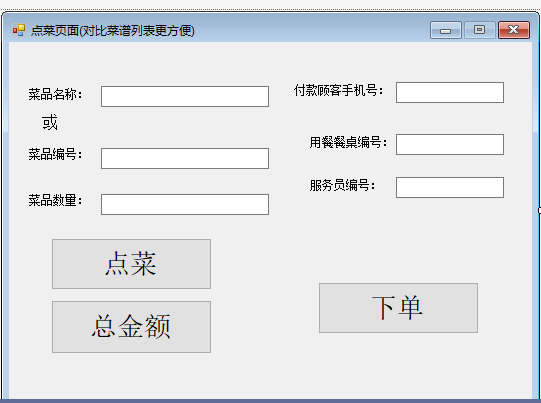


图4-2-6点菜界面

4.2 后端代码实现

我们在下面展示我们的代码。

**4.2.1 C#与SQL Server的连接和交互**

C#是基于.NET平台的，.NET平台提供了许多命名空间，其中允许同本地或远程数据库进行交互式操作时，需要用到的命名空间是ADO. NET，其大部分由System. Data. dll核心程序集来表示，这可以完成应用程序对数据库的不同操作。

ADO. NET支持多种数据源，如果希望与SQL Server进行连接，则需要引入以下命名空间：

using System.Data.SqlClient;

那么我们就可以使用数据提供程序提供的一系列的类来完成与数据库交互的核心功能，一般包括：

（1）sqlconnection类：连接数据库；

（2）executenonquery类：执行sql语句；

（3）getdatatable类：返回填充表格；

（4）executescalar类：查询返回；

以下选取部分代码为例，来说明C#是如何与SQL server进行交互的：

我们定义了一个SQLhelper来存储需要用到的所有连接数据库和执行SQL语句的操作，方便各个界面程序的运行和调试：

public class SqlHelper

{

//连接字符串

private static readonly string connString = ConfigurationManager.ConnectionStrings["connStr"].ConnectionString;

public static object ExecuteScalar(string sql,params SqlParameter[] paras)

{

object o = null;

using (SqlConnection conn = new SqlConnection(connString))

{

//创建Command对象

SqlCommand cmd = new SqlCommand(sql, conn);

//cmd.CommandType = CommandType.StoredProcedure;//存储过程

cmd.Parameters.Clear();

cmd.Parameters.AddRange(paras);

//打开连接

conn.Open(); //最晚打开 最早关闭

o = cmd.ExecuteScalar();//执行查询,返回结果集第一行第一列的值,忽略其他行或列

//关闭连接

//conn.Close();

}

return o;

}

/// <summary>

/// 返回DataTable

/// </summary>

/// <param name="sql"></param>

/// <param name="paras"></param>

/// <returns></returns>

public static DataTable GetDataTable(string sql,params SqlParameter[] paras)

{

DataTable dt = new DataTable();

using (SqlConnection conn = new SqlConnection(connString))

{

//创建Command对象

SqlCommand cmd = new SqlCommand(sql, conn);

//cmd.CommandType = CommandType.StoredProcedure;//存储过程

if (paras!=null)

{

cmd.Parameters.Clear();

cmd.Parameters.AddRange(paras);

}

//打开连接

conn.Open(); //这里打开conn 可以吗? 可以的 后面,da就不会再去关闭

//断开式连接 是不是不用连接数据库呢? 不是

//执行命令 一定是Command来完成

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = cmd;

//SqlDataAdapter da = new SqlDataAdapter(sql, conn);

//打开conn OPened

//数据填充

da.Fill(dt);

//关闭conn

//关闭连接

//conn.Close();

}

return dt;

}

/// <summary>

/// 返回受影响的行数 Insert Update Delete

/// </summary>

/// <param name="sql"></param>

/// <param name="paras"></param>

/// <returns></returns>

public static int ExecuteNonQuery(string sql,params SqlParameter[] paras)

{

int count = 0;

using (SqlConnection conn = new SqlConnection(connString))

{

//创建Command对象

SqlCommand cmd = new SqlCommand(sql, conn);

//cmd.CommandType = CommandType.StoredProcedure;//存储过程

cmd.Parameters.Clear();

cmd.Parameters.AddRange(paras);

//打开连接

conn.Open(); //最晚打开 最早关闭

count = cmd.ExecuteNonQuery();//执行T-SQL语句,返回受影响的行数

//关闭连接

//conn.Close();

}

return count;

}

}

如果登录输入为空值则返回重新输入：

string uName = txtUserName.Text.Trim();

string uPwd = txtUserPwd.Text.Trim();

if(string.IsNullOrEmpty(uName))

{

MessageBox.Show("请输入账号!", "登录提示", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtUserName.Focus();

return;

}

if (string.IsNullOrEmpty(uPwd))

{

MessageBox.Show("请输入密码!", "登录提示", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtUserPwd.Focus();

return;

}

连接数据库判断是否存在相同的账号和密码，如果没有则返回检查账号密码是否错误，否则成功进入主界面：

{

string sql = "select count(1) from UserInfos where UserName=@UserName and UserPwd=@UserPwd";

SqlParameter[] paras =

{

new SqlParameter("@UserName", uName),

new SqlParameter("@UserPwd", uPwd)

};

object o = SqlHelper.ExecuteScalar(sql, paras);

//处理结果

if (o==null||o==DBNull.Value ||((int)o)==0)

{

MessageBox.Show("登录账号或密码有错,请检查!", "登录提示", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

MessageBox.Show("登录成功!", "登录提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

//转到主页面

FrmMain fMain = new FrmMain();

fMain.Show();

this.Hide();

}

我们为一部分关键信息加入了判空提示：

if (string.IsNullOrEmpty(empname))

{

MessageBox.Show("姓名不能为空!", "添加员工提示", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (string.IsNullOrEmpty(Convert.ToString(empnum)))

{

MessageBox.Show("员工编号不能为空!", "添加员工提示", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

对于一些查询提供了两种检索方式。

string dishName= txtDishName.Text.Trim();

string className = txtClassName.Text.Trim();

//

string sql = "select dishnum,dishname,dishprice from manu";

sql += " where 1=1";

if(!string.IsNullOrEmpty(dishName))

{

sql += "and dishName Like @dishname";

}

if(!string.IsNullOrEmpty(className))

{

sql += " and dishnum =@dishnum";

}

SqlParameter[] paras =

{

new SqlParameter("@dishname","%"+dishName+"%"),

new SqlParameter("@dishnum",className)

};

DataTable dtClasses = SqlHelper.GetDataTable(sql,paras);

dgvClassList.DataSource = dtClasses;

}

对于点菜阶段，如果不存在该输入顾客信息则增加该顾客信息，并提示增加成功。如果存在该顾客信息那么就不提示，进入餐桌的状态更改。

string customerTle = txtTel.Text.Trim();

string customerName = txtName.Text.Trim();

string sex = txtSex.Text.Trim();

string paytime = DateTime.Now.ToString();

int tablenum = int.Parse(txttablenum.Text);

int consumAmount = 0;

string sql = "select count(1) from customer where customerTle=@customerTle";

SqlParameter[] paras =

{

new SqlParameter("@customerTle",customerTle),

};

object o = SqlHelper.ExecuteScalar(sql, paras);

if (o != null && o != DBNull.Value && ((int)o) > 0)

{

}

else

{

string sqlAdd = "insert into customer(customerTle,customerName,sex,consumAmount) values(@customerTle,@customerName,@sex,@consumAmount)";

SqlParameter[] parasAdd =

{

new SqlParameter("@customerTle",customerTle),

new SqlParameter("@customerName",customerName),

new SqlParameter("@sex",sex),

new SqlParameter("@consumAmount",consumAmount)

};

int count1 = SqlHelper.ExecuteNonQuery(sqlAdd, parasAdd);

if (count1 > 0)

{

MessageBox.Show($"顾客:{customerName} 添加成功!", "添加顾客提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

对于点菜来说，我们根据菜品的编号从菜谱中检索该菜品的价格，然后在C#中计算该次点菜的总价格：

string dishname = txtName.Text.Trim();

int dishnum = int.Parse(txtNum.Text.Trim());

// string amount = txtAmount.Text.Trim();

int dishamount = int.Parse(txtAmount.Text);

// string sql1 = "select dishnum,dishname,dishprice from manu";

int dishprice = 0;

string sqlprice = "select dishprice from manu where dishnum=@dishnum";

SqlParameter[] paras123 =

{

new SqlParameter("@dishnum",dishnum)

};

object o = SqlHelper.ExecuteScalar(sqlprice, paras123);

dishprice = int.Parse(o.ToString()) ;

int sumprice = dishprice \* dishamount;

SqlParameter[] paras =

{

new SqlParameter("@billnum",billnum),

new SqlParameter("@dishname","%"+dishname+"%"),

new SqlParameter("@dishnum",dishnum),

new SqlParameter("@dishprice",dishprice),

new SqlParameter("@dishamount",dishamount),

new SqlParameter("@sumprice",sumprice)

};

// DataTable dtorder = SqlHelper.GetDataTable(sql1, paras); //先选出菜品

// int count = SqlHelper.ExecuteNonQuery(sql1, paras); //名字和编号

string sqlAdd = "insert into dishinfo (billnum,dishname,dishnum,dishamount,dishprice,sumprice) values (@billnum,@dishname,@dishnum,@dishamount,@dishprice,@sumprice)";

// string sql1 = "insert into dishinfo(dishprice) values (@dishprice) ";

// int count1 = SqlHelper.ExecuteNonQuery(sql1);

int countadd = SqlHelper.ExecuteNonQuery(sqlAdd, paras);

if (countadd > 0)

{

MessageBox.Show($"菜品添加成功!", "添加菜单提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("菜单添加失败!", "添加菜单提示", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

最后根据消费金额可以自动化的更新顾客的历史消费金额：

String hasoccupy = "no";

String sql = "update tableinfo set hasoccupy=@hasoccupy where tablenum=@tablenum";

SqlParameter[] parasAdd11 =

{

new SqlParameter("@hasoccupy",hasoccupy),

new SqlParameter("@tablenum",tablenum)

};

int count123 = SqlHelper.ExecuteNonQuery(sql, parasAdd11);

string sqlAdd1 = "update customer set consumAmount=consumAmount+@actualpay where customerTle=@customerTle";

SqlParameter[] parasAdd1 =

{

new SqlParameter("@customerTle",customerTle),

new SqlParameter("@actualpay",actualpay)

};

int count222 = SqlHelper.ExecuteNonQuery(sqlAdd1, parasAdd1);

if (count222 > 0)

{

MessageBox.Show($"顾客:{customerTle} 已增加此次消费金额!", "修改顾客提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

**4.2.2 SQL Server建立数据库**

根据前文所述的关系模型，建立以下8个基本表（表4-1至表4-8）。

表4-1 billinfo

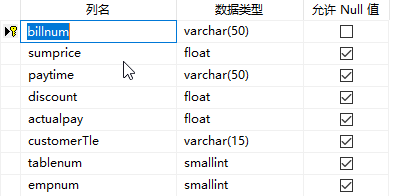


表4-2 customer

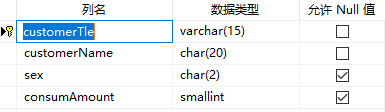


表4-3 dishinfo

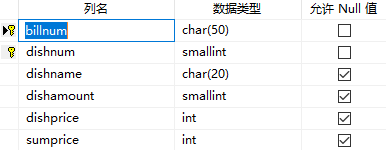


表4-4 empinfo

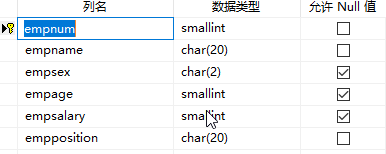


表4-5 manu

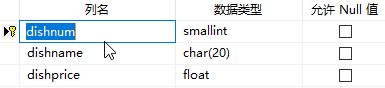


表4-6 order

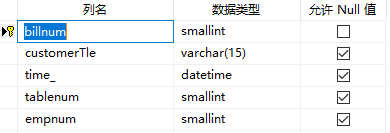


表4-7 tableinfo

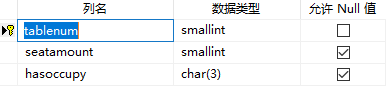
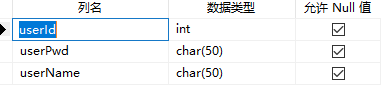


表4-8 Userinfo



数据库关系图如图4-5所示：

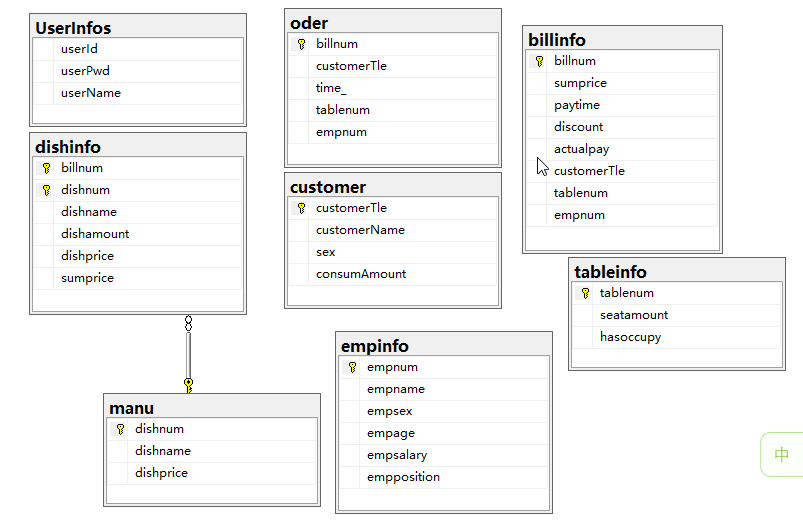


图4-5 数据库关系图

根据数据库逻辑结构设计，编写SQL语句建立数据库。

具体实现代码如下：

/\*建数据库\*/

create database restaurant

/\*建表\*/

create table UserInfos

(

userId int ,

userPwd char(50) ,

userName char(50)

)

insert into UserInfos(userId,userPwd,username) values(1,'123','123')

create table manu

(

dishnum smallint primary key,

dishname char(20)not null,

dishprice float not null

)

create table tableinfo

(

tablenum smallint primary key,

seatamount smallint ,

check (seatamount between 0 and 20),

hasoccupy char(3),

check (hasoccupy in ('yes','no'))

)

create table empinfo

(

empnum smallint primary key,

empname char(20) not null,

empsex char(2)

check(empsex in('男','女')),

empage smallint

check(empage between 20 and 60),

empsalary smallint

check(empsalary between 5000 and 100000),

empposition char(20) not null

)

create table manuinfo

(

billnum char(50),

dishnum smallint,

primary key(billnum,dishnum),

dishname char(20),

dishamount smallint,

dishprice int,

sumprice int

foreign key(dishnum)references manu(dishnum)

)

create table customer

(

customerTle varchar(15) primary key,

customerName char(20)not null,

sex char(2),

check ((sex)in('男','女')),

consumAmount smallint

)

create table billinfo

(

billnum varchar(50) primary key,

sumprice float,

paytime varchar(50),

discount float,

actualpay float,

customerTle varchar(15),

tablenum smallint,

empnum smallint

)

create table oder

(

billnum smallint primary key,

customerTle varchar(15) ,

time\_ datetime,

tablenum smallint,

empnum smallint

)

/\*添加数据\*/

insert into tableinfo(tablenum,seatamount,hasoccupy)

values (1,2,'no'),(2,4,'no'),(3,6,'yes'),(4,2,'no'),(5,4,'no'),(6,6,'yes'),

(7,2,'no'),(8,4,'no'),(9,6,'yes'),(10,2,'no'),(11,4,'no'),(12,6,'yes')

insert into empinfo(empnum,empname,empsex,empage,empsalary,empposition)

values (1,'张三','女',22,6000,'服务员'),(2,'李斯','男',59,20000,'厨师'),(3,'王武','男',35,5000,'服务员'),

(4,'赵柳','女',40,7000,'收银'),(5,'钱六','男',55,8000,'会计')

insert into manu(dishnum,dishname,dishprice)

values(1,'招牌蛋炒饭',10),(2,'鱼香肉丝',15),(3,'大碗牛肉面',9),(4,'可乐鸡翅',15),

(5,'宫保鸡丁',16),(6,'北京烤鸭',50),(7,'可乐',3),(8,'酸辣土豆丝',8),(9,'清水白菜',500),

(10,'双层鸡腿堡',20),(11,'土豆泥',15)

**5 系统测试与应用**

5.1 系统测试运行环境

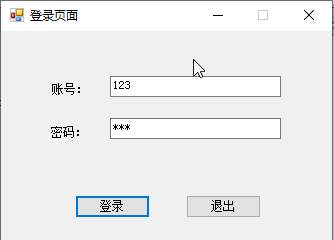
测试系统软件环境：操作系统为Windows 10并且已经安装.Net Framework 4.6.1，已经安装数据库SQL Server 2017和SQL Server Management Studio 17.9.1；

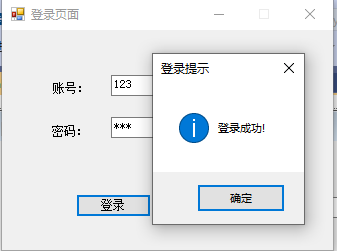
测试系统硬件配置：处理器Intel Core i5-7200U，内存8G，外存256G固态硬盘，显示适配器AMD Radeon R7 M440。

5.2 数据库测试与运行

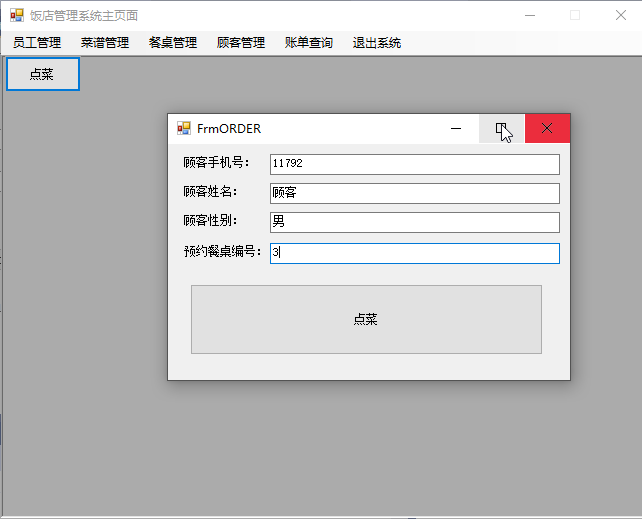
**5.2.1 点菜系统的点菜过程**

**5.2.1.1 用户登录（已有默认账号）**

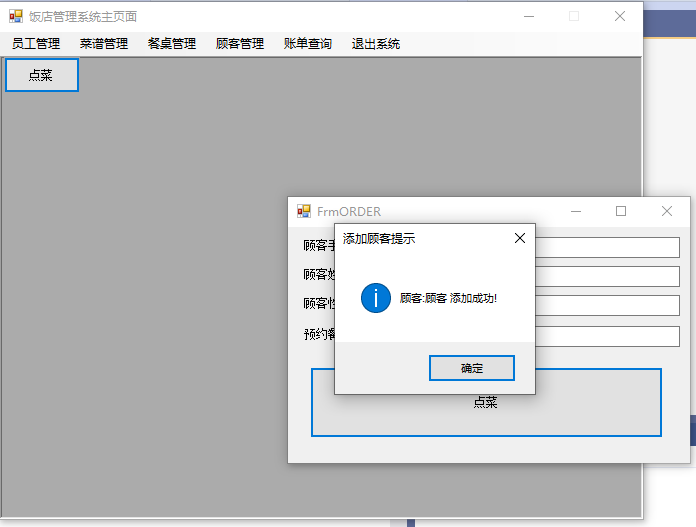




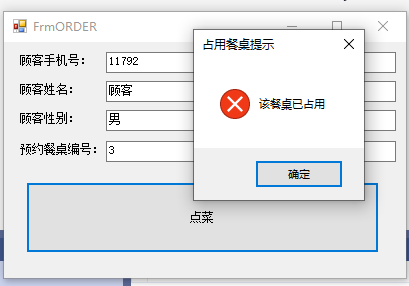
**5.2.1.2**  根据提示输入信息，进入点菜界面

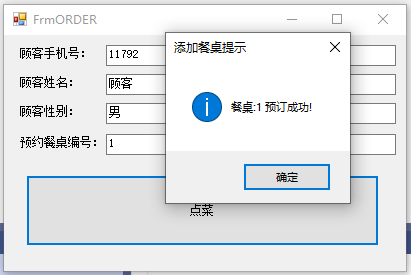


若为新顾客，则自动添加顾客。若为老顾客，直接进入点菜窗口

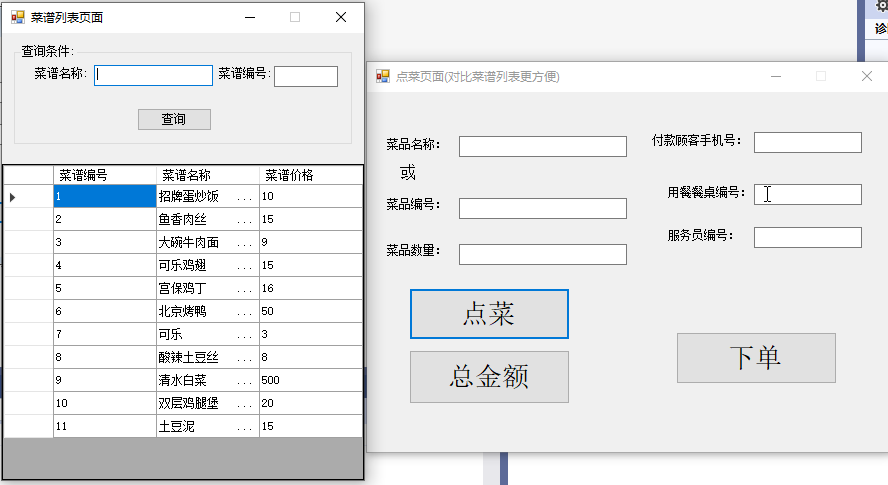


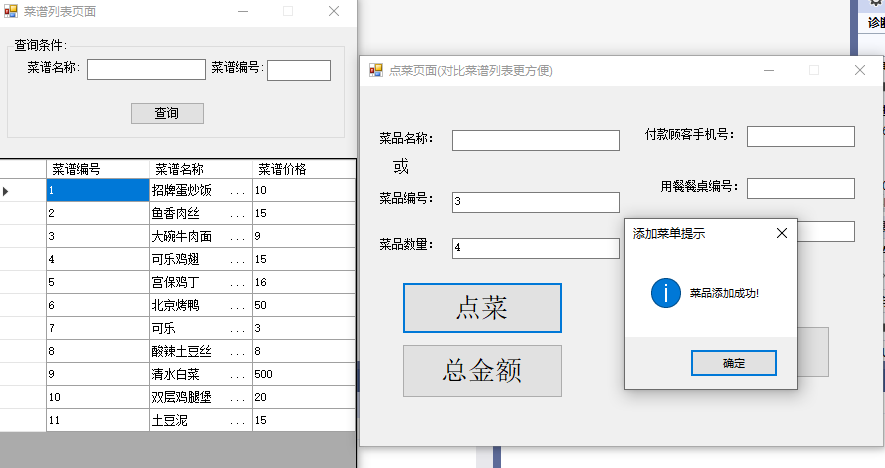
**5.2.1.3 进行餐桌预约**

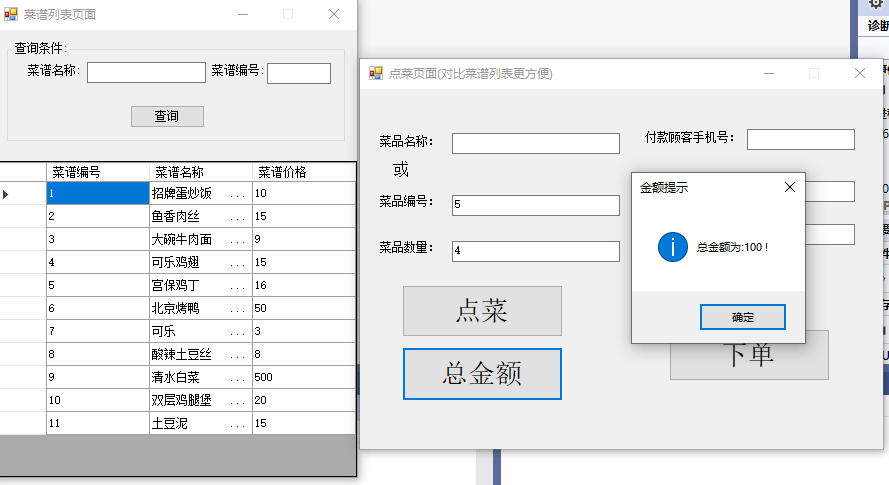
若餐桌已占用，提示已占用： 

选择餐桌空闲，则提示预定成功：

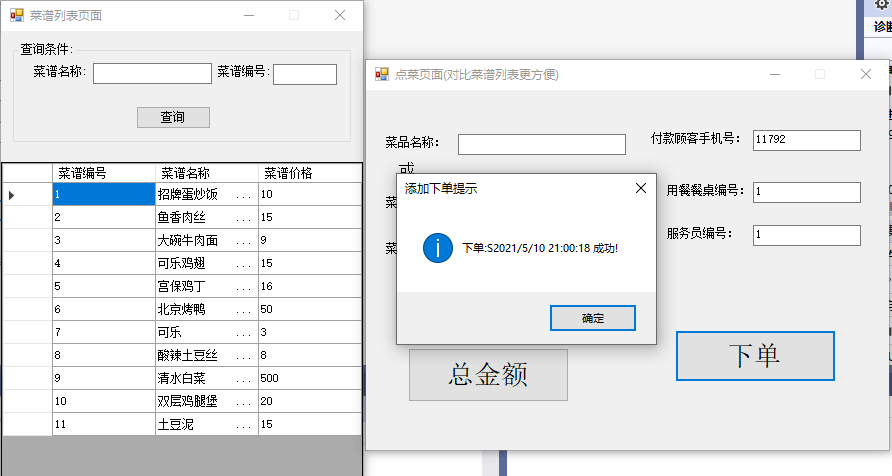
**5.2.1.4 预定成功后根据提示点菜**

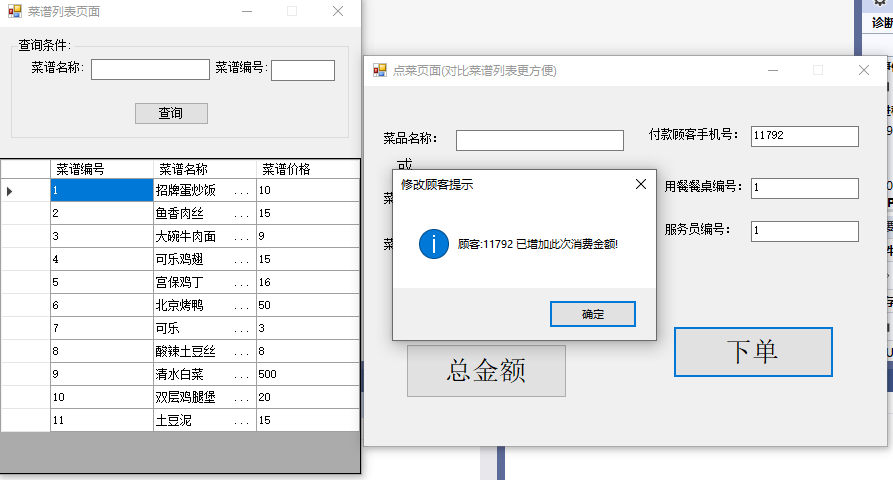


输入菜品编号及数量，即可进行点菜：

点击总金额，可查看当前点菜总金额：

**5.2.1.5 根据提示进行下单付款**



自动增加付款顾客的累计消费金额：

**5.2.2 各列表的查看与修改**

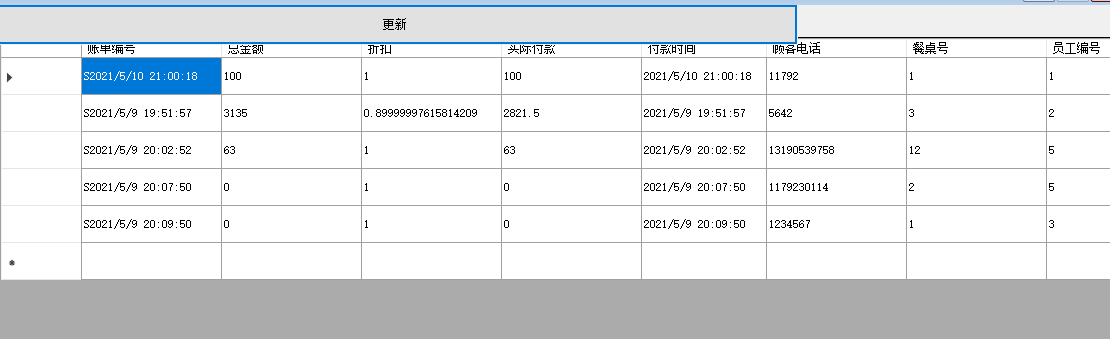
**5.2.2.1 顾客列表查看**

从主界面选择顾客管理—顾客列表，可查看所有消费的顾客：



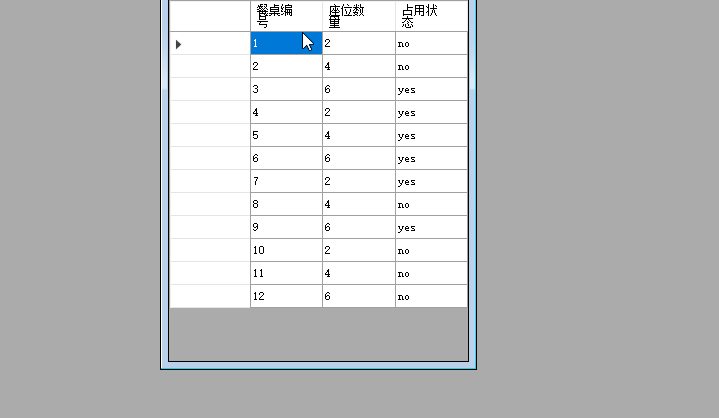
**5.2.2.2 帐单列表查看**

从主界面选择账单查询，可查看账单信息：



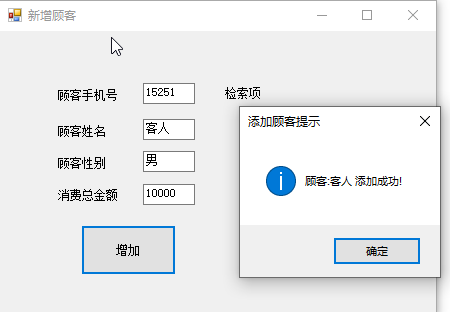
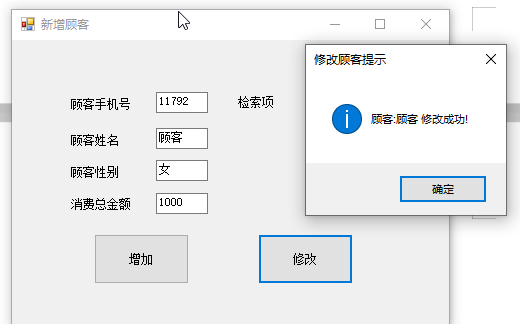
**5.2.2.3 餐桌列表查看**

从主界面选择餐桌管理—餐桌列表，可查看餐桌信息：



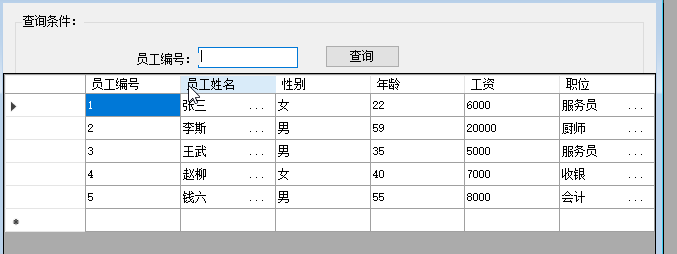
**5.2.2.4 增加或修改顾客信息**

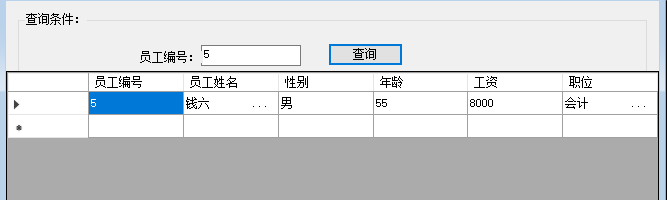
从主界面选择顾客管理—增改顾客，可增加顾客或修改顾客信息：



**5.2.2.5 员工列表查看**

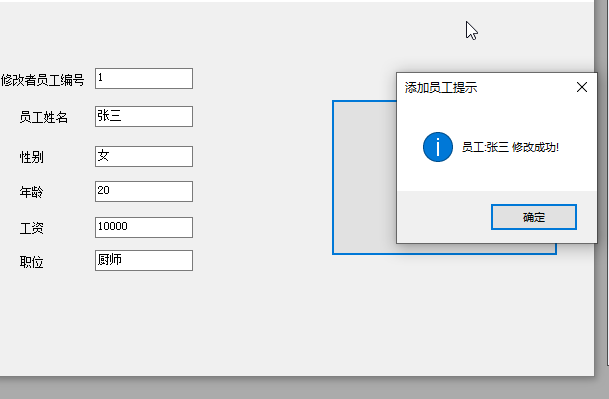
从主界面选择员工管理—员工列表，可查看员工信息（全部员工或检索员工）：

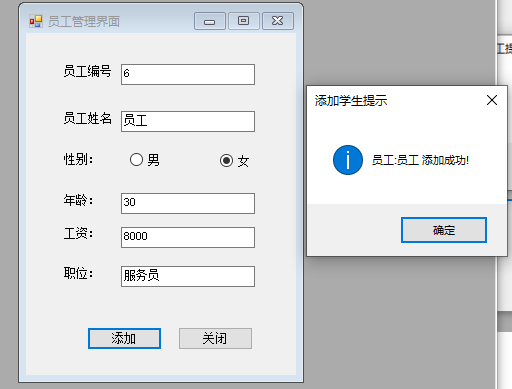




**5.2.2.6 员工列表的修改**

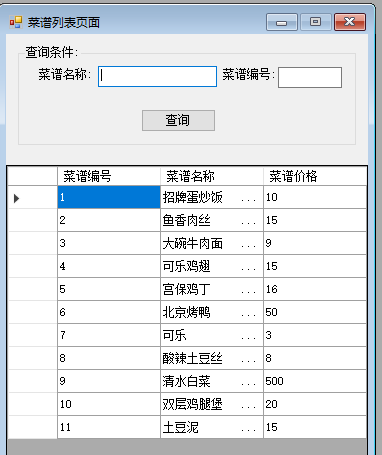
从员工管理中选择新增员工和修改员工，进行员工列表的修改：

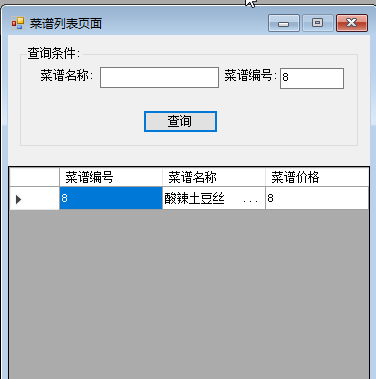




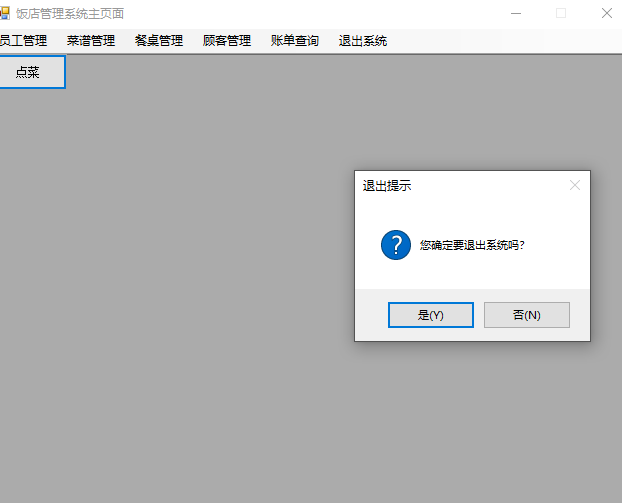
**5.2.2.7 菜品列表查看**

从主界面的菜谱管理—菜谱列表中查看菜谱，查询菜品：





**5.2.2.8 退出系统**

选择退出系统，根据提示退出系统：

**6 总结与展望**

6.1 总结

本系统在着重考察研究当前餐馆点菜和人员管理的基础之上实现了如下核心功能：

1、用户登录功能：管理人员和点餐员用已有的账户进行登录。

2、餐桌预约功能：点餐员进入餐桌管理系统帮顾客预定餐桌。

**3、点菜功能：点餐员根据顾客的需要点击点菜管理进行点菜。**

1. **账单查询功能：点餐员点击账单查询可查询账单并帮助顾客结账。**

**5、人员管理功能：管理员点击人员管理可以对餐馆员工的信息进行人员的变动已经员工信息的修改。**

通过对本课题的研究以及原型系统的设计与开发，在对数据库理论的知识有了更深的理解，能更加熟练地使用SQL语句进行，同时也学习了C#的基本语法并可以使用C#完成基础的软件开发。在开发实践的过程中，本小组成员分明确，积极合作，在交流中互相提高。在开发系统的过程中，我们也出现了许多的问题，但我们的团队会一起查阅资料并且在网上进行相关问题的搜索将问题解决。这次课程设计不仅提高了小组成员的编程能力，也意识到了团队合作的重要性，对软件的开发过程也有了一定的理解和认识。

6.2 不足

目前，本系统的大体核心功能都基本实现，但是整个系统的应用上来说还是有很多的不足，包括：

1. **用户不按指定方式输入会报错。**
2. **数据库只存在于本地，没有实现联网。**
3. **界面不够人性化，界面不够美观。**

6.3 展望

**本系统是面向餐馆的人员管理和点菜的一个系统。希望本系统能加快一些小餐馆的点菜的效率，在人流量较大的时候可以更好的留住顾客。同时也便利管理人员进行人员管理。**

**本系统是对于餐馆点菜管理系统的一个尝试，也仅仅只是一个最基本的样子，希望今后能以本次课程设计为基础，在已有的工作成功之上优化该系统，做到更加人性化，使得界面更加的美观，争取能做到不需要点菜员工进行登录而是让顾客可以通过目前的扫码等方式进行点餐的选择，同时在人员管理的系统上能够为管理员提供更多的数据，更加便利管理员进行人员管理。相信这样的点菜管理系统在未来能够更加完善，从而减少不必要的人力物力。**