长 沙 学 院

软件工程基础实训Ⅱ 实训

|  |  |
| --- | --- |
| **题目** | **基于C/S架构的餐饮管理系统** |
| **学院** | **计算机科学与工程** |
| **专业(班级)** | **软件工程（21软件01）** |
| **姓名** | **邱宇** |
| **学号** | **B20210404122** |
| **指导教师** | **杨刚** |
| **起止日期** | **2023.06.05～2023.06.16** |

实训任务书

**课程名称：软件工程基础实训II**

**实训题目：基于C/S架构的餐饮管理系统**

### 已知技术参数和设计要求：

1. 问题描述（功能要求）：

本次案例将完成一个基于C/S架构的餐饮管理系统。

（1）.进行餐饮管理系统的功能设计。

（2）.设计餐饮管理系统的数据库。

（3）.利用JAVA的GUI类设计开发友好的人机接口。

（4）.利用JAVA语言设计监听用户操作和JDBC进行数据库的处理。

餐饮管理系统主要有6大模块：

（1）.员工管理：对员工实现增删改查。

（2）.客户管理：对客户实现增删改查。

（3）.餐台管理：对餐台实现增删改查。

（4）.菜品管理：对菜品分类、菜品实现增删改查。

（5）.点菜管理：服务员对某客户、某一空闲餐台实行开台，同时实现点菜，将餐台号与所点的菜品对应起来，分别显示出来，并记录开台时间。

（6）.结账管理：收银员对某一餐台通过统计显示消费的菜品清单统计出消费金额，通过手动输入实收金额进行找零的计算，并显示，完成结账的操作，并记录成统计数据。

基础实训II有2周时间40学时，学生基本必做任务：独立完成基于C/S架构的餐饮管理系统。

选做扩展：菜品推荐、销售统计。

**2. 运行环境要求：**

（1）客户端：

windows操作系统，JDK1.6，Idea或Eclipse。

（2）服务器：

MySQL5.6或 SQL Server 数据库服务器。

3. 技术要求：

1） 掌握软件工程的需求分析和系统设计方法。

2) 掌握数据库的分析与设计，完成餐饮管理系统数据库的设计。

3) 掌握JAVA的GUI设计方法，完成登录、主界面、菜品分类管理界面、菜品管理界面等人机接口。

4） 掌握JAVA面向对象程序设计方法，设计系统所需的各种类。

5) 掌握JDBC方法，对数据库进行操作，完成主程序和菜品分类、菜品管理等功能。

### 实训工作量：

40课时

### 工作计划：

1.班级

21软件1-4

2.课时及教室安排

见实训课程工作计划表。

实训报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学生姓名 | 邱宇 | 学号 | B20210404122 | 班级 | 21软件01 |
| 专业 | 软件工程 | | 指导教师姓名 | 杨刚 | |
| 项目名称 | 基于C/S架构的餐饮管理系统 | | | | |
| **实训过程**：（说明实训各阶段的工作过程，包括需求分析、系统设计、编码实现、系统测试、交付实施等阶段的主要工作。）  需求分析：1.能够针对中餐多样化的菜品和特色化的服务提供标准化的管理。  2.能够提供符合餐饮企业自身要求的较科学的标准化、流程化管理，解决餐饮行业专业人才欠缺的  问题。  3.能够针对订餐、点菜、结账等环节的繁重化、复杂化问题，实现强化管理、降低成本、堵漏节流  等。  4.能够针对企业的经营现状做出科学的分析，使企业对市场的应变能力的提高。  系统设计：员工管理：对员工实现CRUD操作。  客户管理：对客户实现CRUD操作。  餐台管理：对餐台实现CRUD操作。  菜品管理：对菜品分类，菜品实现CRUD操作。  业务管理：服务员为某客户对某一空闲餐台进行开台；同时实现点菜，将餐台号与所点的菜品相对  应，分别显示出来，并记录开台时间。  结账管理：收银员对某一餐台通过统计消费的菜品清单统计出消费金额，通过手动输入实收金额进  行找零的计算，并显示找零金额，完成结账的操作，并记录统计数据。 | | | | | |
| **实训成果**：  参见基于C/S架构的餐饮管理系统设计说明书。 | | | | | |
| **实训总结**：（包括心得体会、存在的问题和改进方向。）  一、心得体会  在本次基于C/S架构的餐饮管理系统的实训中，我深刻体会到了系统开发的基本流程和方法。从需求分析、系统设计到编码实现和测试，每一个环节都让我收获颇丰。通过这次实训，我对C/S架构有了更深入的理解，也提升了我的项目实践能力。  首先，需求分析阶段让我学会了如何了解项目的需求并将这些需求转化为清晰、可行的系统功能。同时，我也学会了如何在需求分析过程中进行优先级排序，以确保项目能够按时完成。  其次，系统设计阶段让我掌握了如何根据需求设计合理的系统架构，包括系统界面设计、数据库设计以及后台逻辑设计。在这个过程中，我学会了如何平衡系统的性能、安全性和易用性。  在编码实现阶段，我学会了使用java编程语言和相关技术来实现系统功能。这个过程锻炼了我的编程能力和问题解决能力。  最后，在测试阶段，我学会了如何编写测试用例，对系统进行全面、深入的测试，以确保系统的质量和稳定性。这对我未来的工作和学习都具有重要意义。  二、存在的问题  虽然我在实训中学到了很多知识和技能，但仍然存在一些问题需要改进：  在需求分析阶段，我有时未能充分理解系统的需求，导致后续的开发工作出现困难。今后我会更加努力地弄懂系统需求，确保需求的准确性和完整性。  在系统设计阶段，我需要进一步提高架构设计的灵活性和可扩展性，以适应未来可能出现的新需求和技术变化。  在编码实现阶段，我需要加强代码规范和注释的使用，提高代码的可读性和可维护性。同时，我还需提高自己的编码效率，减少重复劳动。  三、改进方向  为了更好地提升自己在餐饮管理系统的开发过程中，我发现以下几个方面还有待改进：  系统安全性：在开发过程中，我忽视了系统的安全性。今后，我会加强对系统安全的关注，包括用户权限管理、数据加密等方面，以确保系统的安全性和稳定性。  用户体验优化：虽然我在设计阶段已经考虑了用户的易用性，但在实际使用中，仍存在一些不足之处。我将通过收集用户反馈，不断优化系统界面和交互体验，提高用户的满意度。  性能优化：在测试阶段，我发现系统在处理大量并发请求时存在性能瓶颈。为了提高系统的性能，我将在数据库设计、代码优化等方面进行改进，以确保系统能够在高并发场景下正常运行。  四、总结  通过本次基于C/S架构的餐饮管理系统的实训，我对系统开发有了更深入的理解和实践经验。同时，我也意识到自己在需求分析、系统设计、编码实现和测试等方面还存在许多不足。今后，我将继续努力学习 | | | | | |

基于C/S架构的餐饮管理系统

设计说明书

作者：邱宇(B20210404122)

计算机科学与工程学院

2023年 06 月 15日

**摘要**

本文介绍了一个基于C/S架构的餐饮管理系统的设计，包括前台和后台两个部分，以及系统功能模块划分、系统架构设计、项目开发计划、系统测试等方面的内容。该系统将满足餐饮企业的管理需求，提供在线点餐、支付等功能，同时对员工、客户、餐台、菜品等进行管理，采用关系型数据库MySQL保证数据安全和性能优化。在项目开发过程中，将进行单元测试、集成测试、系统测试、压力测试、用户体验测试、安全测试等多方面的测试，确保系统的稳定性和安全性。

**关键词：**java ，C/S架构，餐饮管理系统

目录

[1 引言 1](#_Toc11882504)

[1.1 编写目的 1](#_Toc11882505)

[1.2 参考资料 1](#_Toc11882506)

[2 需求规约 2](#_Toc11882507)

[2.1 功能需求 2](#_Toc11882508)

[2.2 界面需求 3](#_Toc11882509)

[2.3 数据需求 3](#_Toc11882510)

[3 系统设计 4](#_Toc11882511)

[3.1 运行环境 4](#_Toc11882512)

[3.2 系统静态结构设计 4](#_Toc11882513)

[3.3 人机接口设计 5](#_Toc11882514)

[3.4 XXX模块设计说明 7](#_Toc11882515)

[3.5 XXX模块设计说明 7](#_Toc11882516)

[4 数据库设计 8](#_Toc11882517)

[4.1 数据库环境说明 8](#_Toc11882518)

[4.2 数据库的命名规则 8](#_Toc11882519)

[4.3 逻辑结构设计 8](#_Toc11882520)

[4.4 物理结构设计 9](#_Toc11882521)

[4.5 安全性设计 10](#_Toc11882522)

[5 测试用例设计 12](#_Toc11882523)

[附录 14](#_Toc11882524)

[附录1 程序运行结果 14](#_Toc11882525)

[附录2 程序源代码 15](#_Toc11882526)

# 1 引言

## 1.1 编写目的

本文档是基于C/S架构的餐饮管理系统的设计说明书。编写目的是：明确软件需求、概要设计、详细设计和功能测试用例，为软件后续开发和维护提供指导。本文档的读者对象是需求分析人员、系统设计人员、系统开发人员、系统测试人员。

## 1.2 参考资料

[1] 戴远泉. Java高级程序设计实战教程[M]. 北京：人民邮电出版社,2022.

[2] 王珊、萨师煊. 数据库系统概述[M]. 北京：高等教育出版社，2014.

# 2 需求规约

## 2.1 功能需求

餐饮管理系统

结账管理

点菜管理

餐台管理

菜品管理

客户管理

员工管理

图2.1 餐饮管理系统功能模块图

- 员工管理：对员工实现添加、修改、删除、查询等操作。

- 客户管理：对客户实现添加、修改、删除、查询等操作。

- 餐台管理：对餐台实现添加、修改、删除、查询等操作。

- 菜品管理：对菜品分类，菜品实现添加、修改、删除、查询等操作。

- 业务管理：服务员为某客户对某一空闲餐台进行开台；同时实现点菜，将餐台号与所点的菜品相对应，分别显示出来，并记录开台时间。

* 结账管理：收银员对某一餐台通过统计消费的菜品清单统计出消费金额，通过手动输入实收金额进行找零的计算，并显示找零金额，完成结账的操作，并记录统计数据。

2.1.1 用户点菜用例说明

用例名称：用户点菜

用例描述：服务员为某客户对某一空闲餐台进行开台；同时实现点菜，将餐台号与所点的菜品相对应，分别显示出来，并记录开台时间。

前置条件：为用户找一个空着的餐台

后置条件：用户添加菜品

活动步骤：

1. 为用户找一个空着的餐台
2. 系统检查餐台的状态。
3. 系统检查菜品的状态。
4. 系统标记餐台为就餐中状态。
5. 用户添加菜品

异常处理：

1. 如果客户占用正在就餐中的餐台则用例终止。
2. 如果用户新增的菜品不存在，则用例终止。

## 2.2 界面需求

该系统的界面应简洁美观，易于操作，符合人机工程学原则。具体要求如下：

- 用户登录：管理员通过登录管理员账号，对整个系统进行管理。

- 菜单浏览：展示餐厅所有菜品，包括名称、类别、价格、状态等信息。

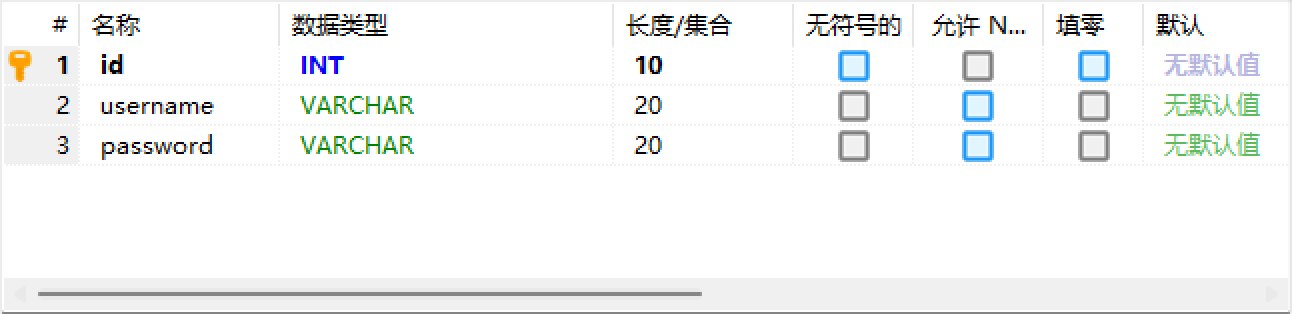
- 菜品详情：提供详细的菜品介绍及营养成分表。

- 在线点餐：顾客可浏览菜单后直接下单。

- 订单状态查询：顾客可查看自己的订单状态，如待确认、已接单、已完成等。

## 2.3 数据需求

用户实体：



员工实体：



客户实体：



餐台实体：



菜品类别实体：



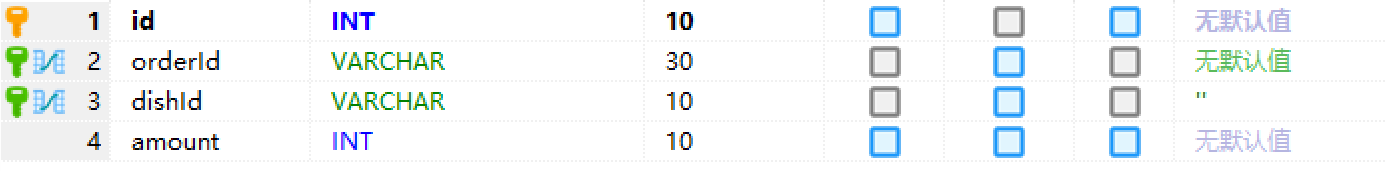
菜品实体：



订单实体：



订单明细实体：



# 3 系统设计

## 3.1 运行环境

该系统的运行环境为Windows，需要安装MySQL数据库、eclipse。

## 3.2 系统静态结构设计

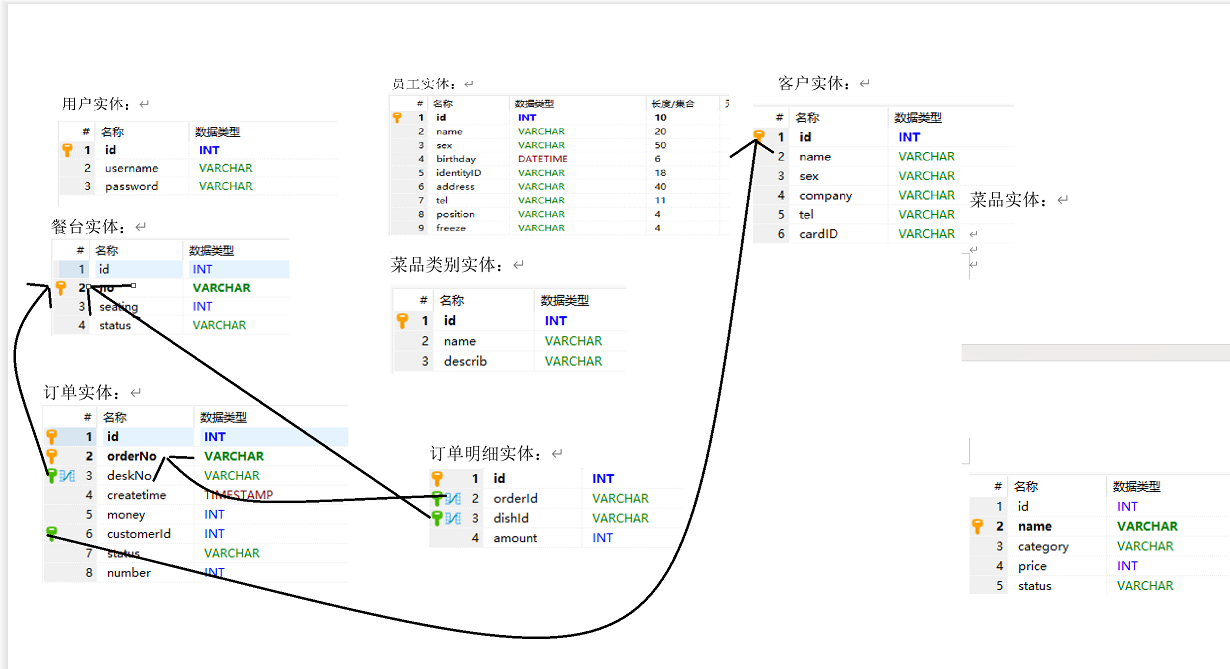


图3.1 餐饮管理系统总体类图

系统静态结构说明

1. 顾客类：该类表示餐厅中的一个顾客，包含顾客的基本信息，如姓名、性别、电话、地址等。在类设计中，需要考虑如何实现顾客的注册、登录、修改个人信息等功能，同时还需要考虑如何保证数据的安全性和完整性，以及如何提高用户体验。

2. 员工类：该类表示餐厅中的一名员工，包含员工的基本信息，如姓名、性别、电话、职位等。在类设计中，需要考虑如何实现员工的注册、登录、修改个人信息等功能，同时还需要考虑如何保证数据的安全性和完整性，以及如何提高用户体验。

3. 餐台类：该类表示餐厅中的一个餐桌或餐位，包含餐桌号、餐位号、所属餐厅等信息。在类设计中，需要考虑如何实现餐桌和餐位的分配和管理，以及如何支持在线预订和点餐等功能。同时还需要考虑如何保证系统的稳定性和可靠性，以及如何提高用户的使用体验。

4. 菜品类：该类表示餐厅中的一道菜品，包含菜名、价格、描述等信息。在类设计中，需要考虑如何实现菜品的分类、搜索和推荐等功能，同时还需要考虑如何保证菜品信息的准确性和及时性，以及如何提高用户的满意度。

5. 订单类：该类表示用户在餐厅中的订单信息，包含订单编号、下单时间、菜品信息等。在类设计中，需要考虑如何实现订单的创建、查询、修改和删除等功能，同时还需要考虑如何保证订单数据的准确性和完整性，以及如何提高用户的使用体验。

综上所述，餐饮管理系统的系统静态结构说明中，针对顾客类、员工类、餐台类、菜品类、订单类等，其在类设计中体现功能需求和非功能需求的方式主要是通过定义属性和方法来实现的。同时还需要考虑如何优化系统的性能和可扩展性，以满足不断变化的业务需求。

## 3.3 人机接口设计

3.3.1 设计应用类

应用类模型中包含了多个边界类，其中包括：

1. 用户类(User):负责处理用户相关的操作，如注册、登录、修改个人信息等。

2. 订单类(Order):负责处理订单相关的操作，如创建订单、查询订单、修改订单等。

3. 菜品类(Meal):负责处理菜品相关的操作，如添加菜品、查询菜品、修改菜品等。

4. 餐台类(Table):负责处理餐桌相关的操作，如添加餐桌、查询餐桌、修改餐桌等。

5. 餐厅类(Restaurant):负责处理餐厅相关的操作，如添加餐厅、查询餐厅、修改餐厅等。

6. 管理员类(Administrator):负责管理整个系统，如添加管理员、查询管理员、修改管理员等。

此外，还可以进一步设计控制类和持久类来实现系统的管理和数据持久化。例如，可以设计一个权限控制类来管理不同用户的访问权限；可以设计一个数据库连接类来管理数据库的连接和断开；可以设计一个数据持久化类来封装数据的持久化操作。

这些控制类和持久类可以通过应用类模型中的接口来与业务逻辑层进行交互，从而实现系统的灵活性和可扩展性。例如，可以通过调用权限控制类的接口来实现不同用户的访问权限控制；可以通过调用数据持久化类的接口来实现数据的持久化和恢复功能。

## 3.4 XXX模块设计说明

1. 用户类(User)

用例1:注册用户

设计考虑：

- 输入合法的用户名、密码、邮箱等信息；

- 检查用户名是否已存在，如果存在则提示错误并要求重新输入；

- 将用户信息保存到数据库中。

用例2:登录用户

设计考虑：

- 输入正确的用户名和密码；

- 检查用户名和密码是否匹配，如果不匹配则提示错误；

- 如果匹配成功，则跳转到主界面。

用例3:修改个人信息

设计考虑：

- 输入正确的用户名；

- 显示用户的个人信息页面；

- 用户可以选择修改姓名、性别、电话、地址等信息；

- 修改后的信息需要进行验证，确保数据的完整性和正确性；

- 将修改后的信息保存到数据库中。

2. 订单类(Order)

需求规格说明：

用例1:创建订单

设计考虑：

- 输入有效的餐桌号、菜品信息、数量等；

- 检查餐桌号是否可用，如果不可用则提示错误；

- 检查菜品是否已售罄，如果已售罄则提示错误；

- 如果所有信息都有效，则将订单信息保存到数据库中。

用例2:查询订单

设计考虑：

- 输入有效的订单编号或用户名；

- 根据订单编号或用户名查询订单信息；

- 如果查询成功，则显示订单的详细信息，包括餐桌号、菜品信息、数量、价格等。

用例3:修改订单状态

设计考虑：

- 输入有效的订单编号和新的订单状态(如“已完成”、“取消”);

- 检查新的订单状态是否合法，如果不合法则提示错误；

- 如果新的订单状态合法，则将订单状态更新到数据库中。

2. 菜品类(Meal)

需求规格说明：

用例1:添加菜品

设计考虑：

- 输入有效的菜品名称、价格、描述等信息；

- 检查菜品名称是否已存在，如果存在则提示错误并要求重新输入；

- 将菜品信息保存到数据库中。

用例2:查询菜品

设计考虑：

- 根据菜品名称或编号查询菜品信息；

- 如果查询成功，则显示菜品的详细信息，包括名称、价格、描述等。

用例3:修改菜品信息

设计考虑：

- 输入正确的菜品编号；

- 显示菜品的基本信息页面；

- 用户可以选择修改菜品名称、价格、描述等信息；

- 修改后的信息需要进行验证，确保数据的完整性和正确性；

- 将修改后的信息保存到数据库中。

# 4 数据库设计

## 4.1 数据库环境说明

本餐饮管理系统的数据库采用MySQL作为关系型数据库管理系统，版本为5.7.28。操作系统为Windows,安装了MySQL服务并进行了相关配置，编程工具为eclipse，版本是 2022-06 (4.24.0)

## 4.2 数据库的命名规则

为了方便管理和维护，本餐饮管理系统的数据库采用了以下命名规则：

* 数据库名称：catering\_management\_system
* 数据表名称：以“order”开头，后接具体的业务字段名，如“order\_table”，“customer\_table”等。
* 字段名：采用驼峰式命名法，首字母小写，单词之间用下划线分隔，如“order\_id”，“customer\_name”等。

## 4.3 逻辑结构设计

1. 用户类(User)

- 关键字：username、password、email、name、gender、phone、address

- 属性：username(varchar)、password(varchar)、name(varchar)、gender(varchar)、phone(varchar)、address(varchar)

- 记录结构：每个用户的信息为一条记录，包括用户名、密码、姓名、性别、电话和地址等属性。

- 文卷结构：一个名为“user”的文卷，包含所有用户的信息。

- 相互关系：无。

2. 餐桌类(Table)

- 关键字：table\_id、status

- 属性：table\_id(int)、status(tinyint)

- 记录结构：每个餐桌的信息为一条记录，包括餐桌号和状态等属性。

- 文卷结构：一个名为“table”的文卷，包含所有餐桌的信息。

- 相互关系：无。

3. 菜品类(Meal)

- 关键字：meal\_id、name、price、description

- 属性：meal\_id(int)、name(varchar)、price(decimal)、description(text)

- 记录结构：每个菜品的信息为一条记录，包括菜品编号、名称、价格和描述等属性。

- 文卷结构：一个名为“meal”的文卷，包含所有菜品的信息。

- 相互关系：无。

4. 订单类(Order)

- 关键字：order\_id、user\_id、table\_id、meal\_id、quantity、status

- 属性：order\_id(int)、user\_id(int)、table\_id(int)、meal\_id(int)、quantity(int)、status(tinyint)

- 记录结构：每个订单的信息为一条记录，包括订单编号、用户ID、餐桌ID、菜品ID、数量和状态等属性。

- 文卷结构：一个名为“order”的文卷，包含所有订单的信息。

- 相互关系：无。

5. 餐厅类(Restaurant)

- 关键字：restaurant\_id、name、address、contact\_info

- 属性：restaurant\_id(int)、name(varchar)、address(varchar)、contact\_info(text)

- 记录结构：每个餐厅的信息为一条记录，包括餐厅编号、名称、地址和联系方式等属性。

- 文卷结构：一个名为“restaurant”的文卷，包含所有餐厅的信息。

- 相互关系：无。

6. 管理员类(Administrator)

- 关键字：admin\_id、username、password、email

- 属性：admin\_id(int)、username(varchar)、password(varchar)、email(varchar)

- 记录结构：每个管理员的信息为一条记录，包括管理员ID、用户名、密码和邮箱等属性。

- 文卷结构：一个名为“admin”的文卷，包含所有管理员的信息。

- 相互关系：无。

## 4.4 物理结构设计

建立系统程序员视图，包括：数据在内存中的安排，包括对索引区、缓冲区的设计；所使用的外存设备及外存空间的组织，包括索引区、数据块的组织与划分；访问数据的方式方法。

4.4.1 数据库表

说明数据库的表的设计。

表4.1 表汇总

|  |  |
| --- | --- |
| 表 名 | 功 能 说 明 |
| User | 系统管理员账号 |
| customer | 客户信息 |
| Employee | 员工信息 |
| Category | 菜品分类信息 |
| Desk | 餐台信息 |
| dishes | 菜品信息 |
| Order | 客户订单详情 |

## 4.5 安全性设计

数据库的安全性设计需要从多个方面考虑，包括访问控制、数据加密、备份和恢复等。以下是一些具体的设计考虑：

1. 访问控制：在数据库的设计中，应该对不同的访问者进行区分，并设置相应的访问权限。例如，可以为管理员和普通用户分别设置不同的访问权限，以确保只有授权的用户才能访问敏感数据。此外，还可以使用角色和组来管理用户的访问权限，以便更好地控制数据的访问。

2. 数据加密：对于敏感数据，可以使用加密技术来保护其安全。例如，可以使用AES等对称加密算法对数据进行加密，或者使用哈希函数对数据进行散列处理。这样即使数据被窃取，也无法直接获取其内容。

3. 备份和恢复：为了防止数据丢失或损坏，应该定期备份数据库，并将备份文件存储在安全的地方。同时，还需要制定应急计划，以便在发生灾难性事件时能够快速恢复数据库。

4. 审计和监控：为了及时发现和处理安全问题，应该对数据库进行审计和监控。例如，可以记录每个用户的操作日志，并对其进行分析和检查。此外，还可以使用安全扫描工具来检测数据库中的漏洞和风险。

总之，数据库的安全性设计需要综合考虑多个因素，包括访问控制、数据加密、备份和恢复、审计和监控等。只有在这些方面都得到充分考虑和实施，才能保证数据库的安全性和保密性。

4.5.1 防止用户直接操作数据库的方法

（1）用户标识和鉴定

（2）存取控制

（3）数据加密

4.5.2 用户帐号密码的加密方法

防止用户直接操作数据库的方法。用户只能用帐号登录到应用软件，通过应用软件访问数据库，而没有其他途径操作数据库。（3）用户帐号密码的加密方法。对用户帐号的密码进行加密处理，确保在任何地方都不会出现密码的明文。

4.5.1 角色与权限

（1）角色与权限。确定每个角色对数据库表的操作权限，如创建、检索、更新、删除等。每个角色拥有刚好能够完成任务的权限，不多也不少。在应用时再为用户分配角色，则每个用户的权限等于他所兼角色的权限之和。

# 5 测试用例设计

对照需求中功能，写出功能的测试用例和测试结果。应考虑边界测试、默认测试、空测试、非法值测试。每个用例应说明下列内容：

(1)测试用例名称

(2)测试项：规定并简要说明本测试用例所要涉及的项和特性、对于每一项、可考虑引用需求说明书和设计说明书。

(3)输入说明：规定执行测试用例所需的各个输入。有些输入可以用值（允许适当的误差）来规定。而另一些输入，如常数表或事务文件可以用名来规定。规定所有合适的数据库、文件、终端信息、内存常驻区域和由操作系统传送的值。规定各输入间所需的所有关系（如时序关系等）。

(4)输出说明：规定测试项的所有输出和特性（如：响应时间）。提供各个输出或特性的正确值（在适当的误差范围内）。

(5)环境要求：规定执行本测试用例所需的硬件特征和配置、系统软件和应用软件。

(6)规程说明：描述对执行本测试用例的测试规程的一切特殊限制。这些限制可以包括特定的准备、操作人员干预、确定特殊的输出和清除过程。

(7)用例间的依赖关系：列出必须在本测试用例之前执行的测试用例名称，归纳依赖性质。

# 附录

## 附录1 程序运行结果

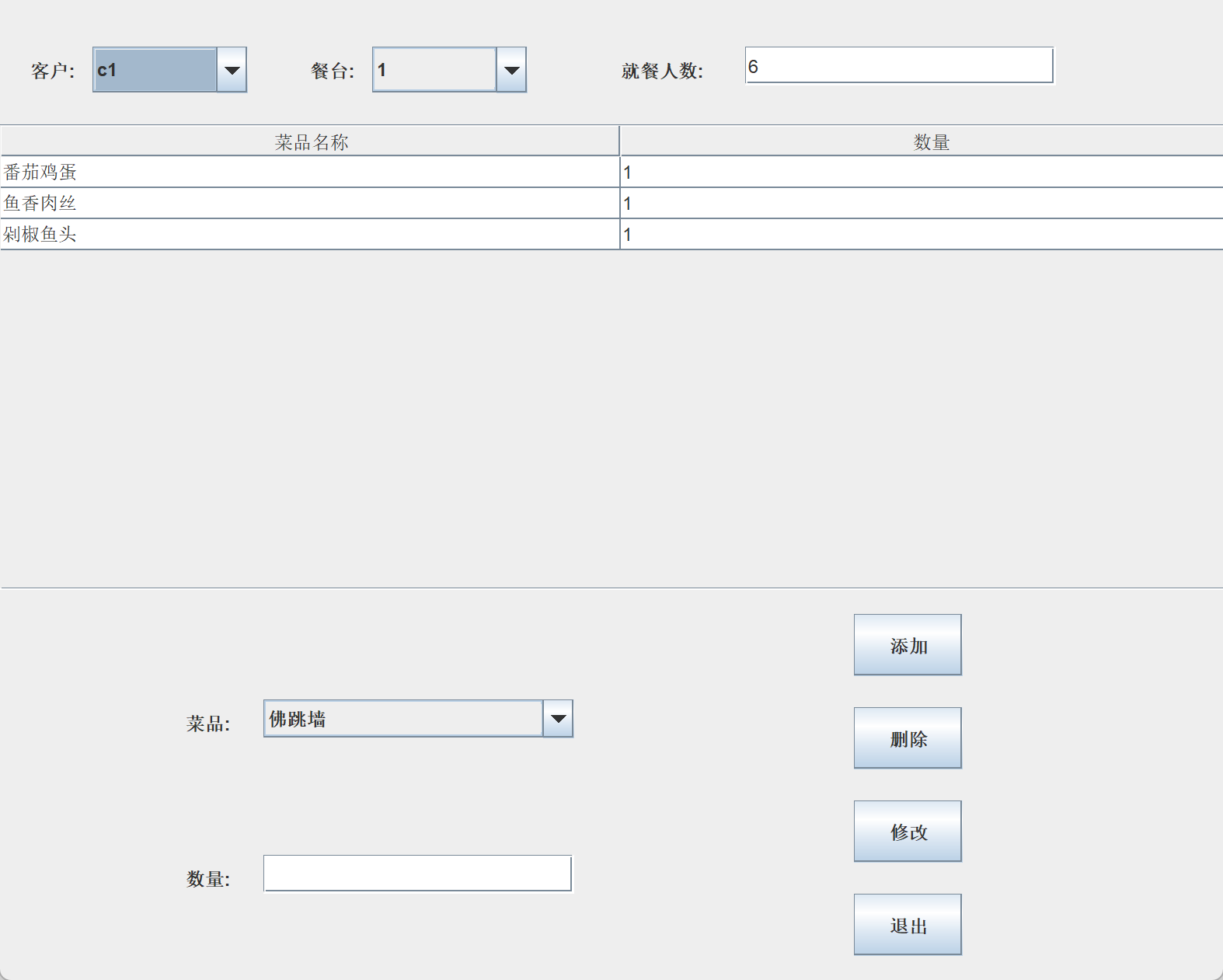


图1 点菜界面

## 附录2 程序源代码

package com.catering.dao;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Timestamp;

import java.util.ArrayList;

import java.util.List;

import com.catering.model.Employee;

import com.catering.model.Order;

public class OrderDAOImpl implements IBaseDAO<Order>{

@Override

public boolean insert(Order or) {

boolean flag = false;

Connection conn = JDBCConnection.getConn();

String sql = "insert into orderr values(?,?,?,?,?,?,?,?)";

try {

PreparedStatement ps = conn.prepareStatement(sql);

ps.setInt(1,or.getId());

ps.setString(2, or.getOrderNo());

ps.setString(3, or.getDeskNo());

ps.setTimestamp(4, or.getCreatetime());

ps.setInt(5,or.getMoney());

ps.setInt(6, or.getCustomeId());

ps.setString(7, or.getStatus());

ps.setInt(8, or.getNumber());

flag = ps.executeUpdate()!=0?true:false;

ps.close();

conn.close();

}catch(Exception e) {

e.printStackTrace();

}

return flag;

}

@Override

public boolean update(Order or) {

boolean flag = false;

Connection conn = JDBCConnection.getConn();

String sql = "update orderr set money=? where id=?";

try {

PreparedStatement ps = conn.prepareStatement(sql);

ps.setInt(1,or.getMoney());

ps.setInt(2, or.getId());

flag = ps.executeUpdate()!=0?true:false;

ps.close();

conn.close();

}catch(Exception e) {

e.printStackTrace();

}

return flag;

}

@Override

public boolean delete(Order or) {

boolean flag = false;

Connection conn = JDBCConnection.getConn();

String sql = "delete from orderr where orderNo=?";

try {

PreparedStatement ps = conn.prepareStatement(sql);

ps.setString(1,or.getOrderNo());

flag = ps.executeUpdate()!=0?true:false;

ps.close();

conn.close();

}catch(Exception e) {

e.printStackTrace();

}

return flag;

}

@Override

public List<Order> selectAll() {

List<Order> orList = new ArrayList<Order>();

Connection conn = JDBCConnection.getConn();

String sql = "select \* from orderr";

try {

PreparedStatement ps = conn.prepareStatement(sql);

ResultSet rs = ps.executeQuery();

while(rs.next()) {

Order or = new Order();

or.setId(rs.getInt(1));

or.setOrderNo(rs.getString(2));

or.setDeskNo(rs.getString(3));

or.setCreatetime(rs.getTimestamp(4));

or.setMoney(rs.getInt(5));

or.setCustomeId(rs.getInt(6));

or.setStatus(rs.getString(7));

or.setNumber(rs.getInt(8));

orList.add(or);

}

rs.close();

ps.close();

conn.close();

}catch(Exception e) {

e.printStackTrace();

}

return orList;

}

public static void main(String[] args) {

}

}