**数据库应用实训**

**课程大作业**

|  |  |
| --- | --- |
| **作业题目：** | **外卖管理系统数据库的设计与实现** |
| **任课老师：**  **学生姓名：** |  |
|  |
| **学 院 ：** | **网络安全与信息技术学院** |
| **班 级 ：** | **计科21-2** |
| **学 号：** | **211181510** |
| **时 间：** | **2024年6月1日** |

[**摘 要：** 2](#_Toc170241077)

[**第一章 绪论** 3](#_Toc170241078)

[**1.1 研究背景及意义** 3](#_Toc170241079)

[**1.2 研究动机** 3](#_Toc170241080)

[**1.3 相关技术** 3](#_Toc170241081)

[**1.3.1 编程语言** 3](#_Toc170241082)

[**1.3.2 GUI库** 3](#_Toc170241083)

[**1.3.3 数据库技术** 3](#_Toc170241084)

[**1.3.4 日期和时间处理** 3](#_Toc170241085)

[**1.4 研究目标与贡献** 3](#_Toc170241086)

[**第二章 需求分析** 4](#_Toc170241087)

[2.1 可行性分析 4](#_Toc170241088)

[**2.1.1 技术可行性** 4](#_Toc170241089)

[**2.1.2 经济可行性** 4](#_Toc170241090)

[**2.1.3操作可行性** 4](#_Toc170241091)

[**2.2 需求分析** 4](#_Toc170241092)

[**2.2.1 系统功能需求分析** 4](#_Toc170241093)

[**2.2.2 系统性能需求分析** 5](#_Toc170241094)

[**第三章 系统总体设计** 6](#_Toc170241095)

[**3.1 系统总体结构设计** 6](#_Toc170241096)

[**3.2 系统模块功能介绍与设计** 6](#_Toc170241097)

[**3.3 数据库设计** 7](#_Toc170241098)

[**3.3.1 数据库需求分析** 7](#_Toc170241099)

[**3.3.2 数据库概念结构设计** 7](#_Toc170241100)

[**3.3.3 数据库逻辑结构设计** 9](#_Toc170241101)

[**第四章 系统运行与测试** 11](#_Toc170241102)

[**4.1系统运行环境** 11](#_Toc170241103)

[**4．2 系统实现** 11](#_Toc170241104)

[**4.3系统功能视图** 15](#_Toc170241105)

[**4.4 测试结果与分析** 17](#_Toc170241106)

[**第五章 总结与展望** 18](#_Toc170241107)

[**5.1研究总结** 18](#_Toc170241108)

[**5.2存在的问题** 19](#_Toc170241109)

[**5.3未来研究方向** 19](#_Toc170241110)

[**参考文献：** 20](#_Toc170241111)

[**致谢** 21](#_Toc170241112)

**外卖管理系统数据库的设计与实现**

姓名：郭裕龙

（网络安全与信息技术学院计算机科学与技术2021-2班，新疆 伊宁市 835000）

摘 要**：**

随着电子商务的蓬勃发展，外卖服务已成为现代生活的一个重要组成部分。为了满足市场对高效、便捷外卖服务的需求，本文设计并实现一个基于Python语言和MySQL数据库的外卖管理系统。该系统采用B/S架构，通过PyCharm开发环境进行编码，为客户端和商家端提供了一系列的功能。系统后端采用Python语言，利用其强大的库支持和简洁的语法特性，提高了开发效率。前端界面则采用通过Tk库构建的GUI界面友好、响应迅速，极大地提升了用户的交互体验。数据库层面，MySQL的使用为系统提供了稳定可靠的数据存储解决方案。 在系统测试阶段，通过模拟用户操作和压力测试，验证了系统的稳定性和响应速度。测试结果表明，系统能够处理高并发请求，满足实际应用场景的需求。

**关键词：**外卖管理系统；Python；MySQL；B/S架构；客户端；商家端

**第一章 绪论**

**1.1 研究背景及意义**

随着信息技术的飞速发展，互联网已经深入人们的日常生活，极大地改变了人们的生活方式。特别是在餐饮服务领域，外卖服务以其便捷性和高效性迅速成为现代人生活中不可或缺的一部分。根据市场研究数据，外卖市场的规模在过去几年里持续扩大，用户基数不断增长，展现出巨大的市场潜力和发展空间。该系统需要能够提供简洁直观的用户界面、高效的订单处理能力，以帮助商家提高运营效率，同时为用户提供更加个性化和便捷的服务体验。本研究旨在开发一个基于Python语言和MySQL数据库的外卖管理系统，具有以下几方面的重要意义：**提升运营效率，优化用户体验，支持小型商家，促进技术应用，提供研究基础。**

**1.2 研究动机**

当前市场上的外卖系统存在功能复杂、操作繁琐的问题，尤其对于中小商家来说，使用门槛较高。因此，本研究旨在开发一个简洁、易用的外卖管理系统，方便用户和商家进行日常操作。

**1.3 相关技术**

**1.3.1 编程语言**

Python：作为一种高级编程语言，Python因其简洁、易读和强大的库支持而广泛应用于各种软件开发中。

**1.3.2 GUI库**

Tkinter：Python的标准GUI库，用于创建跨平台的图形用户界面。

**1.3.3 数据库技术**

MySQL：一个流行的开源关系型数据库管理系统，用于存储和管理数据。mysql.connector：Python的MySQL驱动程序，用于连接和操作MySQL数据库。

**1.3.4 日期和时间处理**

datetime：Python内置的日期和时间处理库，用于获取当前时间戳。

**1.4 研究目标与贡献**

本文开发的外卖管理系统包含客户端和商家端两个视图，并通过Tkinter图形界面库和MySQL数据库实现了系统的前后端连接。

**第二章 需求分析**

2.1 可行性分析

**2.1.1 技术可行性**

**编程语言和框架：**使用Python语言，结合Tkinter库开发GUI，以及mysql.connector进行数据库操作，这些技术都是成熟且广泛使用的，确保了技术的可行性。**数据库技术：**采用MySQL数据库，它是一个稳定且功能丰富的关系型数据库管理系统，适用于处理结构化数据。**系统兼容性：**Tkinter跨平台特性保证了系统在Windows、macOS和Linux上的兼容性。**第三方库支持：**Python有丰富的第三方库支持，便于扩展功能，如使用pandas进行数据分析，使用numpy进行科学计算等。

**2.1.2 经济可行性**

* **开发成本**：Python和Tkinter是开源免费的，MySQL也提供了免费的社区版，这大大降低了开发成本。
* **维护成本：**Python社区活跃，遇到问题容易找到解决方案，降低了后期维护成本。
* **硬件投资：**系统对硬件要求不高，可以在常规的服务器或个人计算机上运行，减少了硬件投资。

**2.1.3操作可行性**

Tkinter开发的GUI简单直观，易于操作，适合不同层次的用户。由于界面友好，用户培训成本较低，用户可以快速上手。系统操作流程设计合理，用户可以轻松完成注册、下单、查询等操作。

**2.2 需求分析**

**2.2.1 系统功能需求分析**

**外卖系统功能需求分析着重于构建一个用户友好、高效且安全的服务平台。系统应允许用户轻松注册和管理个人信息，浏览和搜索多样化的商品选项，并能够方便地下单和支付。同时，商家需要能够管理自己的商品列表，包括添加新商品、更新价格和库存，以及处理订单。订单管理是系统的核心，需要支持从下单到配送的整个流程，包括订单状态的实时更新和通知。此外，系统还应提供查询功能，使用户和商家能够轻松追踪订单和交易历史。安全性也是至关重要的，系统必须确保所有用户数据和交易都是安全的，通过实施加密和安全协议来保护用户隐私。最后，为了支持未来的增长和变化，系统的设计需要具备良好的可扩展性和灵活性，以适应不断变化的市场需求和技术进步。**

**2.2.2 系统性能需求分析**

* **响应时间：**系统操作的响应时间应不超过几秒钟，保证用户体验。
* **并发用户数**：系统应能支持一定数量的并发用户操作，具体数量根据预期用户规模确定。
* **数据准确性：**系统处理的数据必须准确无误，特别是在订单处理和财务统计方面。
* **系统稳定性：**系统需要**7×24**小时稳定运行，保证服务的连续性。
* **扩展性：**系统设计应考虑未来可能的功能扩展，如增加新的支付方式、配送服务等。

**第三章 系统总体设计**

**3.1 系统总体结构设计**

本外卖管理系统采用B/S（Browser/Server）架构，客户端通过浏览器访问系统，服务器端负责处理业务逻辑和数据库交互。系统总体结构设计包括前端展示层、业务逻辑层和数据访问层。前端展示层：使用tkinter库构建图形用户界面，为用户提供直观的操作界面。业务逻辑层：以Python语言实现，处理系统的核心业务逻辑，如订单处理、用户管理等。数据访问层：通过mysql.connector与MySQL数据库进行交互，实现数据的增删改查操作。

**3.2 系统模块功能介绍与设计**

系统主要分为两大模块：客户端和商家端，每个模块包含多个子功能。​

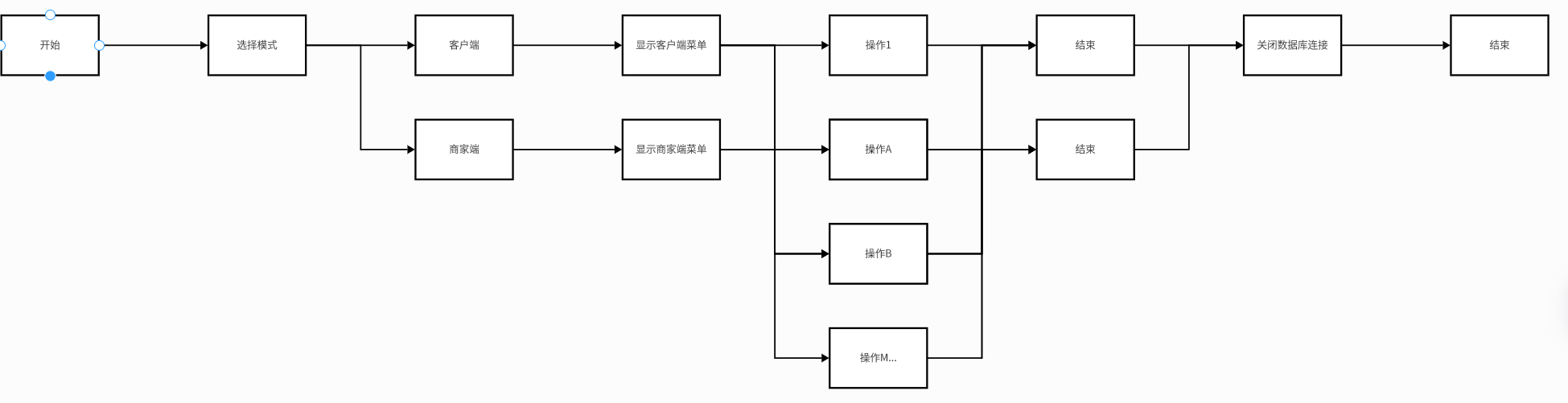


图3.2 系统功能结构图

* 客户端模块：
  + 查看商品信息：展示可购买的商品列表。
  + 查看商家信息：展示出商家的店铺相关信息。
  + 下单操作：允许用户选择商品并下单。
  + 订单查询：查看自己的订单状态。
  + 注册信息：用户注册新账户进入购买。
  + 取消订单与确认订单：客户对订单的操作。
* 商家端模块：
  + 商家注册：允许新的商家注册账户。
  + 商品录入：商家对新的商品进行录入账户。
  + 商品信息修改：商家可以修改自己的商品。
  + 查看订单信息：商家对所有顾客订购的商品进行查看。
  + 查看商家信息：查看自己的门店信息。
  + 接单：接单订单状态。
  + 查看营业额：统计和展示商家的营业额。

**3.3 数据库设计**

数据库设计是系统设计的重要组成部分，它直接影响到系统的运行效率和数据的一致性。

**3.3.1 数据库需求分析**

根据系统的功能需求，数据库需要存储以下信息：用户信息：包括客户和商家的基本信息。商品信息：包括商品名称、价格、库存等。订单信息：记录订单的详细数据，如商品ID、数量、总价等。营业额统计：记录商家的营业额数据。

**3.3.2 数据库概念结构设计**

通过E-R图来表示数据实体之间的关系，主要实体有：

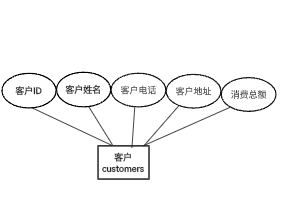


图3.1客户（Customers）

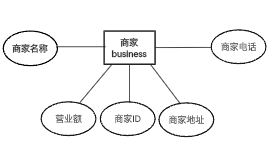


图3.2商家（Businesses）

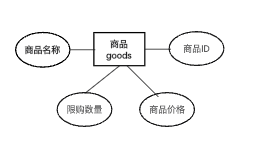


图3.3商品（Goods）

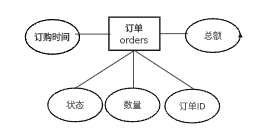


图3.4订单（Orders）

实体之间的关系如下：

一个商家可以有多个商品，一个商品属于一个商家。

一个客户可以有多个订单，一个订单属于一个客户。

一个订单包含一个商品。

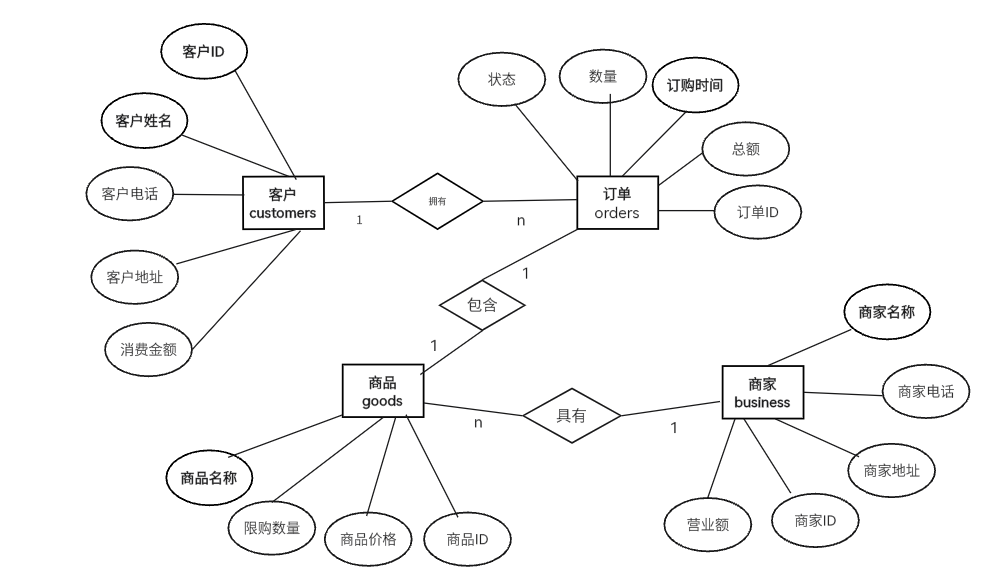


图3.5总E-R图

**3.3.3 数据库逻辑结构设计**

数据库的逻辑结构通过表和字段来实现，主要表结构如下：

Customers表(存储客户ID、姓名、电话、地址、消费金额)。

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| Id | INT | 客户编号 |
| Name | VARCHAR(255) | 客户姓名 |
| Phone | VARCHAR(0) | 客户电话 |
| Address | VARCHAR(255) | 客户地址 |
| Total\_spent | DECIMAL(10,2) | 消费金额 |

表3.6 客户表（Customers）

Businesses表(存储商家ID、名称、电话、地址、营业额等)。

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| Id | INT | 商家编号 |
| name | VARCHAR(255) | 商家名称 |
| Phone | VARCHAR(255) | 商家电话 |
| Address | VARCHAR(255) | 商家地址 |
| Total\_revenue | DECIMAL(10,2) | 营业额 |

表3.7 商品表（Businesses）

Goods表(存储商品ID、名称、商家ID、价格、限购数量等)。

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| Id | INT | 商品编号 |
| name | VARCHAR(255) | 商品名称 |
| Business\_id | INT | 商家编号 |
| price | DECIMAL(10,2) | 商品价格 |
| Limit\_num | INT | 限购数量 |

表3.8 商品表（Goods）

Orders表(存储订单ID、客户ID、商品ID、订单时间、数量、总价、订单状态等)。

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| Id | INT | 订单编号 |
| Order\_time | DATETIME | 订单时间 |
| Customer\_id | INT | 客户编号 |
| Goods\_id | INT | 商品编号 |
| Total\_price | DECIMAL(10,2) | 总价 |
| State | INT | 订单状态 |

表3.9 订单表（Orders）

表之间通过外键来维护关系，确保数据的引用完整性。

**第四章 系统运行与测试**

**4.1系统运行环境**

系统运行在Python 3.8环境下，依赖于Tkinter库进行图形界面开发，MySQL数据库进行数据存储与管理。

**4．2 系统实现**

数据库连接配置

db\_config = {

'user': 'root',

'password': 'g1234',

'host': 'localhost',

'database': 'km'

}

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor()

# 用于显示欢迎信息的标签

welcome\_label = tk.Label(root, text="欢迎使用外卖系统", font=("Arial", 16))

# Client View

def client\_view():

client\_root = Tk()

client\_root.title("客户端")

messagebox.showinfo("提示", "你已进入客户端视图")

# Business View

def business\_view():

business\_root = Tk()

business\_root.title("商家端")

messagebox.showinfo("提示", "你已进入商家端视图")

**创建触发器**

触发器一：#限购以及计算订单总额 限购以及计算订单总额的触发器  
 CREATE TRIGGER limit\_buy  
 BEFORE INSERT  
 ON orders  
 FOR EACH ROW  
 BEGIN  
 DECLARE p DOUBLE(16,2); /\*商品单价\*/  
 DECLARE lim INT; /\*限购数量\*/  
 SET p=(SELECT price FROM goods WHERE goods.id=NEW.goods\_id);  
 SET lim=(SELECT limit\_num FROM goods WHERE goods.id=NEW.goods\_id);  
 if (lim IS NOT NULL AND NEW.number>lim) THEN  
 SET NEW.number=lim;  
 END IF ;  
 SET NEW.total\_cost=NEW.number\*p;  
 END ;#  
  
触发器二：#更新客户消费总额 在客户提交订单之后，自动将消费金额累加到客户总的消费金额里。  
CREATE TRIGGER calculate\_consumption  
 AFTER INSERT  
 ON orders  
 FOR EACH ROW  
 BEGIN  
 UPDATE customers  
 SET consumption=consumption+NEW.total\_cost  
 WHERE customers.id=NEW.customer\_id;  
 END ;  
  
触发器三：#更新商家总营业额以及退款  
#商家接单后，自动将订单金额累加到商家的营业额里；  
#客户取消订单，自动将客户总消费减去本次订单金额。  
CREATE TRIGGER calculate\_turnover  
 BEFORE UPDATE  
 ON orders  
 FOR EACH ROW  
 BEGIN  
 IF (NEW.state=1 AND OLD.state=0) THEN  
 UPDATE businesses  
 SET turnover=turnover+OLD.total\_cost  
 WHERE id=(SELECT business\_id FROM goods WHERE goods.id=NEW.goods\_id);  
 ELSEIF (NEW.state=3 AND OLD.state=0) THEN  
 UPDATE customers  
 SET consumption=consumption-OLD.total\_cost  
 WHERE id=OLD.customer\_id;  
 ELSEIF (NEW.state<OLD.state) THEN  
 SET NEW.state=OLD.state;  
 END IF ;  
 END ;

#查看触发器  
SHOW TRIGGERS;

**创建视图**

举例创建订单详情视图 ：

CREATE VIEW order\_details AS  
SELECT orders.id AS 订单编号,  
businesses.name AS 商家名称,  
goods.name AS 商品名称,  
number AS 数量,  
total\_cost AS 总额,  
state AS 订单状态,  
customers.name AS 客户姓名,  
customers.phone AS 电话,  
customers.address AS 地址,  
order\_time AS 时间  
FROM ((businesses INNER JOIN goods ON businesses.id = goods.business\_id)  
INNER JOIN orders ON  
goods.id = orders.goods\_id)  
INNER JOIN customers ON  
orders.customer\_id=customers.id;

#查看视图  
SELECT \* FROM order\_details;

**为表创建索引**

# 为orders表的customer\_id和goods\_id创建索引  
CREATE INDEX idx\_orders\_customer\_id ON orders(customer\_id);  
CREATE INDEX idx\_orders\_goods\_id ON orders(goods\_id);  
  
# 为goods表的business\_id和price创建索引  
CREATE INDEX idx\_goods\_business\_id ON goods(business\_id);  
CREATE INDEX idx\_goods\_price ON goods(price);  
  
# 为businesses表的id和name创建索引  
CREATE INDEX idx\_businesses\_id ON businesses(id);  
CREATE INDEX idx\_businesses\_name ON businesses(name);  
  
# 为customers表的id和name创建索引  
CREATE INDEX idx\_customers\_id ON customers(id);  
CREATE INDEX idx\_customers\_name ON customers(name);  
  
# 如果经常根据下单时间查询订单，为order\_time创建索引  
CREATE INDEX idx\_orders\_order\_time ON orders(order\_time);

**4.3系统功能视图**

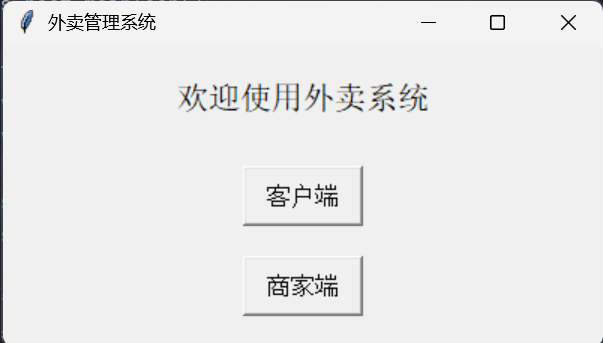
****

图4.1 系统页面

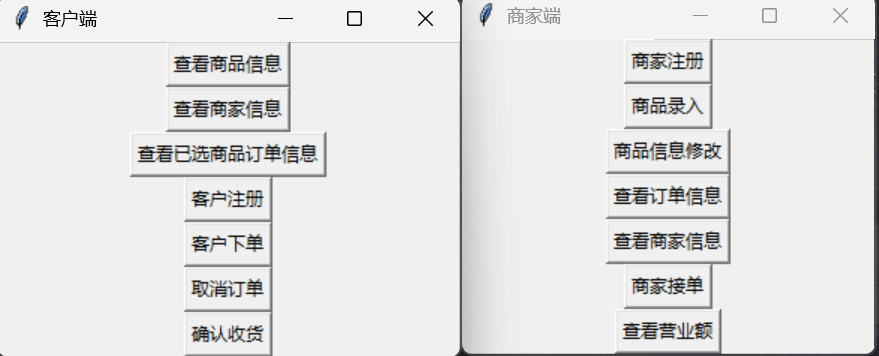
****

图4.2进入客户端和商家端界面

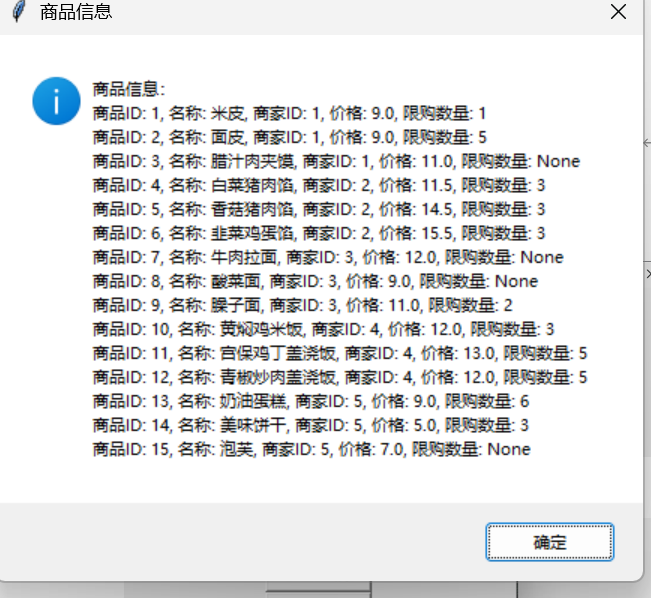
****

图4.3客户查看商品信息

****

图4.4客户查看商家信息

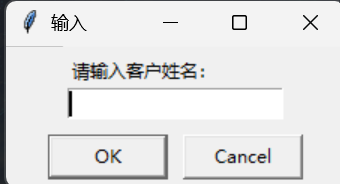


图4.5客户注册

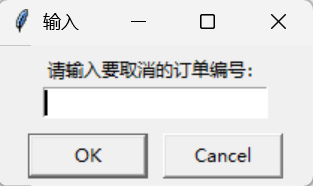


图4.6客户订单的取消

**4.4 测试结果与分析**

1测试环境

• 操作系统：Windows 10

• Python 版本：3.8

• MySQL 版本：8.0

• GUI 库：Tkinter

2测试用例概览

• 数据库连接测试：验证系统能否成功连接到MySQL数据库。

• 客户注册测试：测试客户信息是否能够正确注册到数据库。

• 商家注册测试：测试商家信息是否能够正确注册到数据库。

• 商品录入测试：测试商品信息是否能够正确录入数据库。

• 订单创建测试：测试订单创建流程是否顺畅，订单信息是否正确存储。

• 订单状态更新测试：测试订单状态（接单、取消、确认收货）更新是否成功。

• 信息查询测试：测试系统是否能够正确查询并显示客户、商家、商品和订单信息。

• 营业额计算测试：测试商家营业额计算功能是否准确。

3测试结果

• 数据库连接测试：结果：成功连接到数据库。分析：数据库配置正确，连接函数实现正常。

• 客户注册测试：结果：客户信息注册成功，但在输入无效数据时，系统未能给出明确的错误提示。分析：注册功能实现正常，但需要增强输入验证和错误处理。

• 商家注册测试：结果：同客户注册测试。分析：同客户注册测试。

• 商品录入测试：结果：商品信息录入成功，限购数量留空时，数据库中该字

NULL。分析：商品录入功能正常，能够处理可选字段。

• 订单创建测试：结果：订单创建成功，但未进行客户存在性验证。分析：需要在

建订单前验证客户是否存在。

• 订单状态更新测试：结果：订单状态更新成功，但系统在更新不存在的订单时没

给出错误提示。分析：需要增强订单存在性验证和错误处理。

• 信息查询测试：结果：所有查询功能均返回预期结果。分析：查询功能实现正常，能够正确从数据库检索信息。

• 营业额计算测试：结果：计算结果准确，但测试覆盖不足，未能验证日期范围和状态过滤。分析：需要增加更多边界条件和异常情况的测试。

4总结

• 系统的主要功能实现正常，能够完成基本的数据库操作和信息管理。

• 需要增强输入验证和错误处理，确保系统的健壮性。

• 需要增加更多的测试用例，特别是对于边界条件和异常情况的测试

**第五章 总结与展望**

**5.1研究总结**

本系统旨在提供一个全面的解决方案，以满足现代外卖业务的需求。它实现了外卖管理的基本功能，包括客户和商家的注册与信息管理、商品信息管理、订单管理等。通过Tkinter提供了简洁易用的用户界面，使用MySQL数据库保证了数据的可靠存储与查询。

**5.2存在的问题**

目前系统功能较为基础，一些高级功能如推荐系统、数据分析等尚未实现。提供的外卖管理系统代码实现了基本的客户和商家管理、商品信息维护、订单处理等功能，但存在一些潜在的问题和改进空间。代码的结构较为线性，缺乏模块化，这可能影响未来的维护和扩展。错误处理和用户输入验证机制需要加强，以提供更精确的错误反馈并防止潜在的安全风险。数据库交互方面，直接的用户输入可能导致SQL注入，应采用参数化查询来提高安全性。系统在处理高并发请求时的性能和稳定性也需进一步考量。用户界面设计可能需要更复杂的布局和交互，以适应更高级的UI需求。此外，系统的性能优化、安全性措施、测试流程的完善、可扩展性设计，以及法律遵从性都是需要进一步关注和改进的方面。尽管系统满足了基本的业务需求，但为了提高其健壮性、效率和用户友好度，仍需在多个层面进行优化和增强。

**5.3未来研究方向**

未来的研究方向针对外卖管理系统可以聚焦于提升用户体验、增强系统性能、加强安全性、引入智能化服务、增强可扩展性与集成能力、优化多平台兼容性、拓展国际化和本地化支持、以及深化数据分析和业务洞察等方面。通过精心设计的用户界面改进，可以使用户交互更为直观和友好，从而提高用户满意度。系统性能可以通过采用高效的算法、优化数据库查询和引入缓存机制等方式得到提升。安全性增强可以通过实现更先进的用户认证和授权机制以及强化数据加密来实现。智能化服务的引入，比如利用机器学习为用户提供个性化推荐，将进一步提升服务的质量和效率。此外，系统架构的灵活性对于支持新功能的集成至关重要，同时，移动端应用的开发和响应式设计将确保系统在不同设备上的兼容性。对多语言和本地化的支持将帮助系统进入更广阔的市场。深入的数据分析将为商家提供宝贵的业务洞察，帮助他们优化运营策略。这些研究方向将共同推动外卖管理系统向更高效、更智能、更用户友好的方向发展。

**参考文献：**

[1] 王珊, 萨师煊. 数据库系统概论（第5版）[M]. 高等教育出版社, 2014.

[2] 郑庆华, 郭春雨. 电子商务概论（第3版）[M]. 清华大学出版社, 2019. 325-337 .

[3] 赵一鸣. Python高级编程（第2版）[M]. 机械工业出版社, 2021. 159-176.

[4] 王亚强. MySQL数据库性能优化与管理[M]. 电子工业出版社, 2017.

[5] 李强. 基于MySql的设计与实现[J]. 西安文理学院学报：自然科学版, 2017(2):50-54.

[5] 张华. 数据库系统实现[M]. 清华大学出版社, 2018. 342-352.

**致谢**

在此论文完成之际，我要向所有在这个过程中给予我帮助和支持的人们表示由衷的感谢。首先，衷心感谢我的指导老师，在整个课程大作业期间给予我的悉心指导和宝贵建议，您的专业知识和耐心指导不仅帮助我克服了许多技术难题，也让我在学术研究和项目开发方面受益匪浅。其次，我要感谢我的同学和朋友们，在作业中与我分享经验和建议，帮助我不断改进和完善项目。你们的支持和鼓励是我坚持不懈的动力源泉。同时，我要感谢学校为我提供的良好学习环境和丰富的资源，在项目开发过程中，学校的图书馆、实验室和计算设施为我的研究提供了有力的支持。此外，感谢家人对我的理解和支持，你们的关心和鼓励使我能够全身心地投入到学习和项目中，并顺利完成这项课程大作业。最后，感谢所有在我学习和项目开发过程中给予帮助和支持的老师、同学及朋友们，正是因为有了你们的帮助，我才能克服各种困难，顺利完成此次大作业。再次向大家表示诚挚的感谢！