徽标

AI 生成的内容可能不正确。

**基于 Python Flask 与 SQL Server 的饭店点餐系统课程设计说明书**

|  |  |  |
| --- | --- | --- |
| **学** | **院** | 理学院 |
| **专** | **业** | 数据科学与大数据技术 |
| **班** | **级** | 大数据232 |
| **学** | **号** | 1230510006 |
| **姓** | **名** | 张方波 |

**摘要**

本文设计并实现了一个基于 Python Flask 框架与 SQL Server 数据库的饭店点餐系统。系统后端采用 Flask 微框架，前端使用 HTML5 与 Jinja2 模板进行页面渲染，数据持久化使用 SQL Server，通过 pyodbc 驱动实现高效稳定的数据库连接与操作。系统具备用户登录、菜单浏览、添加购物车、退菜、折扣结账、历史订单查询等核心功能，并对输入进行了完整的校验与错误处理。在系统测试阶段，通过功能测试、性能测试和安全测试，验证了系统的稳定性、响应速度和安全性。最终，系统表现出良好的可扩展性与可维护性，为餐饮行业的信息化提供了切实可行的解决方案。

**关键词：** Flask；SQL Server；点餐系统；系统设计；数据库联动

目录

[1 绪论 4](#_Toc203173253)

[**1.1** **研究背景** 4](#_Toc203173254)

[**1.2** **研究现状** 4](#_Toc203173255)

[**1.3** **研究目标** 4](#_Toc203173256)

[**2 相关技术介绍** 4](#_Toc203173257)

[**2.1 Python 与 Flask** 4](#_Toc203173258)

[2.2 SQL Server 与 pyodbc 5](#_Toc203173259)

[**2.3 Jinja2 模板引擎** 5](#_Toc203173260)

[3 需求分析 5](#_Toc203173261)

[**3.1 功能需求** 5](#_Toc203173262)

[**3.2 非功能需求** 6](#_Toc203173263)

[4 系统设计 6](#_Toc203173264)

[4.1 总体架构 6](#_Toc203173265)

[4.2 数据库设计 6](#_Toc203173266)

[**4.2.1 设计原则** 6](#_Toc203173267)

[**4.2.2 表结构** 7](#_Toc203173268)

[**4.3 功能模块划分** 8](#_Toc203173269)

[**5 系统实现** 8](#_Toc203173270)

[**5.1 后端实现** 8](#_Toc203173271)

[**5.2 前端实现** 10](#_Toc203173272)

[**6 系统测试** 11](#_Toc203173273)

[**7 总结与展望** 13](#_Toc203173274)

[8. 参考文献 14](#_Toc203173275)

# 1 绪论

## **研究背景**

随着互联网技术的飞速发展，餐饮行业逐步向数字化、智能化转型。传统的人工点餐方式效率低、耗时长，且易发生信息丢失与统计错误。基于 Web 的在线点餐系统可以实现菜品信息实时展示、订单在线下单和自动统计结算，极大提升了点餐与结算效率。

## **研究现状**

目前市场上主流的点餐系统大多依赖移动端小程序或大型电商平台，部署与维护成本较高。本设计旨打造一个轻量级、本地化部署的点餐系统，满足中小型饭店对后端灵活性和数据库定制化的需求，同时保持良好的用户体验和系统性能。

## **研究目标**

* 构建基于 Flask 的后端服务，提供清晰的路由与数据接口
* 设计 SQL Server 数据库表结构，实现高效数据存储与查询
* 实现用户身份验证与会话管理，保证系统安全
* 完成点餐、退菜、折扣结账以及历史订单查询等业务流程
* 对系统进行全面测试，确保系统稳定、可用

# **2 相关技术介绍**

## **2.1 Python 与 Flask**

Python 作为一门简洁高效的编程语言，拥有丰富的第三方库和活跃的社区，适合 Web 应用开发。Flask 是一个轻量级的 Web 微框架，特点如下：

* **路由管理**：通过装饰器快速定义 URL 与视图函数映射
* **模板渲染**：集成 Jinja2 模板引擎，实现动态页面生成
* **会话支持**：内置会话机制，可弹性扩展为数据库或缓存存储
* **扩展丰富**：大量插件可用于数据库 ORM、安全认证、文件上传等

## 2.2 SQL Server 与 pyodbc

SQL Server 是微软推出的企业级关系型数据库管理系统，具有以下优势：

* **事务支持**：ACID 特性保证数据一致性
* **复杂查询**：支持存储过程、视图、函数与索引优化
* **高可用**：内置镜像、复制与 Always On 可用性组

pyodbc 是 Python 下连接 ODBC 数据源的通用驱动，能够与 SQL Server 无缝集成，支持批量操作和游标控制。

## **2.3 Jinja2 模板引擎**

Jinja2 是 Flask 默认的模板引擎，支持：

* 变量替换、条件判断、循环等模板语法
* 过滤器与宏定义，方便代码复用
* 模板继承，构建公共页面布局

# 3 需求分析

## **3.1 功能需求**

1. **用户登录与权限管理**：
   * 页面输入用户名与密码
   * 验证用户合法性，并在会话中记录用户 ID
   * 未登录用户禁止访问核心业务页面
2. **菜单浏览与分页展示**：
   * 从 Menu 表中获取菜品信息
   * 支持分页展示、按类别或价格区间筛选
   * 每个菜品显示名称、图片（可扩展）、单价
3. **添加购物车**：
   * 用户可单击“加入购物车”按钮，向会话中添加或更新菜品数量
   * 购物车页面展示所选菜品、单价、数量与小计
4. **退菜与数量修改**：
   * 在购物车中，可修改数量或将数量设为 0 移除菜品
   * 实时更新小计与总价
5. **折扣结账**：
   * 输入折扣率（0.1～1.0），支持空值默认 1.0
   * 计算总价、折后实付金额，四舍五入保留两位小数
   * 插入 Orders 与 OrderItems 表，记录订单汇总与明细
6. **历史订单查询**：
   * 列出当前用户所有历史订单，按时间倒序
   * 展示每笔订单的汇总信息与菜品明细
7. **输入校验与错误处理**：
   * 前端使用 HTML5 校验，后端补充边界检查
   * 捕获数据库异常，友好提示用户

## **3.2 非功能需求**

* **性能**：使用游标批量插入与事务提交，保证高并发环境下性能
* **安全**：防止 SQL 注入，使用参数化查询
* **可维护性**：模块化代码结构，统一数据库连接管理
* **可扩展性**：可集成 Redis、Vue 前端等组件

# 4 系统设计

## 4.1 总体架构

系统采用典型的 MVC 架构：

* **Model**：pyodbc 访问 SQL Server，封装数据访问层函数
* **View**：Jinja2 模板负责页面展示与用户交互
* **Controller**：Flask 路由函数接收请求，完成业务逻辑后渲染模板或重定向

## 4.2 数据库设计

### **4.2.1 设计原则**

* **规范化**：避免重复数据，使用外键维护关联
* **可扩展**：预留菜品分类、用户角色等字段
* **高性能**：对高频查询字段建立索引（如 created\_at）

### **4.2.2 表结构**

|  |  |  |
| --- | --- | --- |
| **表名** | **字段及类型** | **说明** |
| users | id INT PK IDENTITY, username NVARCHAR(50), password NVARCHAR(255) | 登录认证表 |
| Menu | id INT PK IDENTITY, name NVARCHAR(100), price DECIMAL(10,2) | 菜品信息表 |
| Orders | id INT PK IDENTITY, customer\_name NVARCHAR(50), total DECIMAL(10,2), discount DECIMAL(5,2), final\_price DECIMAL(10,2), created\_at DATETIME | 订单汇总表 |
| OrderItems | id INT PK IDENTITY, order\_id INT FK→Orders(id), menu\_item\_id INT FK→Menu(id), quantity INT | 订单明细表 |

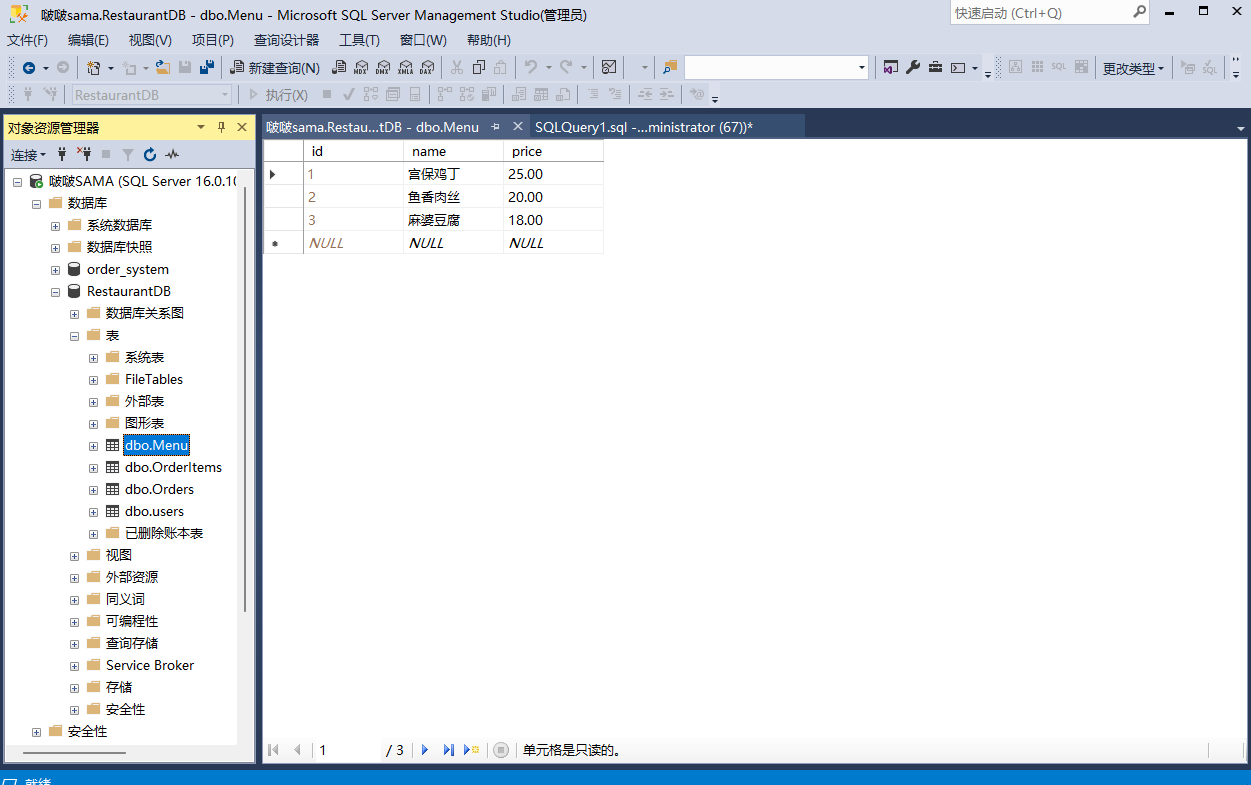


图1.sql server表格设计

## **4.3 功能模块划分**

* **登录模块**（/login）：表单提交→参数化查询→会话写入
* **菜单模块**（/order, /add\_to\_cart）：查询 Menu，会话中维护购物车字典
* **购物车模块**（/cart, /update\_cart）：从会话拉取购物车，渲染页面并更新数量
* **结账模块**（/checkout, /result）：计算金额→事务插入两张表→清空购物车→结果显示
* **历史订单模块**（/history）：按用户过滤订单→多表 JOIN 获取明细→模板渲染

# **5 系统实现**

## **5.1 后端实现**

后端主要由 Flask 路由与数据库交互模块组成，核心代码可拆分为以下几个子模块：

**1.数据库连接与工具函数**

文本

AI 生成的内容可能不正确。

图2.数据库连接

* 使用 pyodbc 驱动进行 ODBC 连接，采用 Windows 集成身份验证。
* 返回 (conn, cursor) 以便在各路由中重用。

文本

AI 生成的内容可能不正确。

图3.转换

将原生 cursor.fetchall() 返回的元组列表，转换为字典列表，方便模板中通过 item['name'] 访问字段。

**2.用户认证模块** (/login)

文本, 应用程序

AI 生成的内容可能不正确。

图4.用户认证

* 使用参数化查询防止 SQL 注入。
* 登录成功后将用户信息保存在 session 中。

**3.菜单展示与购物车维护** (/order、/add\_to\_cart、/cart、/update\_cart)

* **菜单查询**：从 Menu 表中拉取 id, name, price，并传递给 order.html 模板。
* **加入购物车**：session['cart'] 维护为 {item\_id: qty} 结构，每次点击“加入购物车”即累加数量。
* **查看购物车**：根据 session['cart'] 中的键，批量查询对应菜品，计算 subtotal = price \* qty，并累加 total。数据类型使用 Decimal 精确计算。
* **更新购物车**：遍历提交表单中带 quantity\_ 前缀的字段，重新构建购物车字典，实现退菜或数量修改。

4.**结账与订单持久化** (/checkout)

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

图5.汇总表

* 使用事务一次性写入订单和明细，确保数据一致性。
* 对 discount 字段做非空与格式校验。

5.**历史订单查询** (/history)

文本

AI 生成的内容可能不正确。

图6.历史查询

主表与明细表 JOIN 查询，构造嵌套字典列表，为模板提供完整订单结构。

以上模块组合构成了后端业务逻辑的核心，所有数据库操作均采用参数化 API，保证安全性；金额运算使用 decimal.Decimal 实现高精度计算；会话管理使用 Flask 内置 session，无须额外依赖。

## **5.2 前端实现**

* **login.html**：使用 <form method="post">，required 与 type="password" 确保输入
* **order.html**：循环 items 列表，展示菜名、价格与“加入购物车”按钮
* **cart.html**：可直接修改 <input type="number"> 数量并提交到 /update\_cart
* **checkout.html**：折扣输入框 step="0.01" min="0.1" max="1.0"，默认值 1.00
* **result.html**：显示 order\_id，并提供跳转
* **history.html**：循环 orders，内层循环 order\_items，展示历史记录

# **6 系统测试**

通过在浏览器打开<http://127.0.0.1:5000>来进行系统的使用

**用例覆盖**：登录成功/失败、购物车增删改、结账有/无折扣、历史订单查询

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图7.登陆界面

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图8.点餐界面

图形用户界面, 表格

AI 生成的内容可能不正确。

图9.购物车界面

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图10.结账确认界面

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图11.订单确认界面

文本, 电子邮件

AI 生成的内容可能不正确。

图12.查看历史订单界面

# **7 总结与展望**

本课程设计实现了一个结构清晰、功能齐全的点餐系统，代码约 300 行，易于维护与扩展。未来可集成 Vue 前端，支持移动适配；引入 Redis 缓存，提升并发性能；集成第三方支付接口，实现完整电商体验。

# 参考文献

 李明, 王芳. 基于 Flask 的在线点餐系统设计与实现. 软件导刊, 2019, 18(5): 45–50.

 张强, 刘洋. 基于 Python 的餐饮管理系统研究与开发. 计算机应用研究, 2020, 37(12): 3658–3662.

 陈伟, 赵丽. 基于 Django 与 Vue 的电子商务系统架构与实践. 电子技术应用, 2019, 45(8): 112–116.

 Kumar A., Singh R. Web‑Based Restaurant Management System Using MVC Pattern. International Journal of Advanced Computer Science and Applications, 2018, 9(4): 123–129.

 Sharma P., Garg R. Design and Implementation of Online Food Ordering System. Proceedings of the 2020 IEEE International Conference on Web Services, 2020: 28–35.