**一、引言**

1.1 目的

本文档旨在描述"锦食江安"食堂点单小程序的系统架构与设计方案，为开发团队提供详细的技术指导。文档涵盖系统整体架构、模块划分、数据设计及人机交互方案，适用于以下读者：

* 开发人员：依据本文档进行代码实现
* 测试人员：基于设计规范制定测试用例
* 项目管理人员：监控开发进度与质量
* 食堂运营方：理解系统技术实现原理

1.2 范围

本系统是基于微信生态的食堂点餐解决方案，主要包含：

1. **用户端小程序**：提供在线点单、人流量查询、评价互动等功能
2. **管理端后台**：实现订单管理、菜品配置、数据分析等运营功能
3. **数据中台**：处理实时人流量计算、智能推荐等核心算法

项目目标：

* 缩短用户平均就餐等待时间至5分钟以内
* 提升食堂档口资源利用率30%
* 建立数字化反馈闭环（评价响应率≥90%）

1.3 文档结构

本文档共分为8个主要章节：

1. 引言：说明文档目的与范围
2. 系统概述：总体功能与设计理念
3. 系统架构：技术栈与模块划分
4. 数据设计：数据库结构与字典
5. 组件设计：核心模块实现方案
6. 人机交互：界面设计规范
7. 需求追踪：需求与设计对应关系
8. 附录：补充技术细节

1.4 参考文献

* 《软件需求规格说明书》(V2.0)，项目组
* 《微信小程序开发规范》(v1.3)，腾讯公司
* 《Spring Boot技术白皮书》(2.6版本)，Pivotal
* 《MySQL数据库设计指南》(8.0版本)，Oracle

1.5 术语与缩写

* **QPS**（Queries Per Second）：每秒查询率，用于衡量系统吞吐量的关键性能指标
* **AES-256**：采用256位密钥的高级加密标准，用于敏感数据的存储与传输加密
* **RESTful**：基于HTTP协议的API设计风格，遵循资源化与无状态原则
* **SLA**（Service Level Agreement）：服务等级协议，例如系统可用性99.9%的保障承诺
* **POI**（Point of Interest）：兴趣点数据，在本系统中特指食堂窗口的地理位置信息

**二、系统概述**

食堂点单小程序是针对高校食堂就餐场景开发的独立应用，旨在通过数字化手段提升就餐效率和用户体验。

该系统与微信平台集成，利用微信的用户基础和支付功能，实现无缝对接。系统开发完成后，将与食堂现有的管理流程相结合，提升食堂的运营管理水平。

2.1 功能全景

系统采用"前后端分离+微服务"架构，主要功能模块包括：

**用户侧功能**

* 在线点单与支付：支持用户浏览菜品、加入购物车、在线支付（微信支付）。
* 实时人流量查询：展示食堂各区域的实时人流量，帮助用户避开拥堵。
* 智能推荐：基于用户历史订单和热门菜品，自动推荐新品或高匹配度菜品。
* AI客服：支持自然语言处理，解答用户关于菜品、食堂窗口、营业时间等问题。

#### 2.2 运行环境

前端：微信小程序平台，支持iOS和Android操作系统。

后端：基于Springboot微服务架构，使用Java开发。

数据库：MySQL用于存储业务数据

#### 2.3 设计和实现约束

遵循微信小程序开发规范和接口标准。

后端系统需具备高并发处理能力，支持至少10用户同时在线。

系统需支持多食堂管理，覆盖不同校区或楼层的食堂数据。

#### 2.4 假设和依赖关系

假设微信平台的稳定性和安全性能够满足小程序的运行需求。

依赖第三方支付接口（微信支付）和地图API（用于食堂位置展示）的稳定性和兼容性。

食堂管理方提供必要的菜品信息和运营数据。

三、系统架构

#### 

#### 3.1 架构设计

本系统采用分层架构设计，将系统划分为多个模块化的子系统，以实现系统的完整功能。这种架构设计有助于提高系统的可维护性、可扩展性和可测试性。以下是系统的高层职责划分及其子系统：

#### 3.1.1 系统职责划分

前端子系统：负责提供用户交互界面，包括用户端的微信小程序和管理端的Web后台管理系统。

后端子系统：负责处理业务逻辑，包括订单管理、数据监控、菜品推荐、AI客服等功能。

数据层子系统：MySQL数据库负责数据的存储和管理。

第三方服务子系统：负责与外部服务集成，如微信支付、地图API、AI服务等。

#### 3.1.2 子系统协作

各子系统通过定义清晰的接口进行交互，确保系统的整体功能得以实现。前端子系统通过HTTP请求与后端子系统通信，后端子系统处理业务逻辑后，从数据层子系统获取或存储数据。第三方服务子系统通过API与后端子系统集成，提供支付、地图、AI等服务。

#### 3.1.3 架构图

系统架构图如下所示：



#### 3.2 分解说明

#### 

#### 3.2.1 子系统分解

以下是系统的主要子系统及其功能描述：

1. 前端子系统

用户端（微信小程序）：

提供用户界面，支持菜品浏览、点单、支付、人流量查询等功能。

使用微信小程序框架开发，支持iOS和Android设备。

2. 后端子系统

订单管理模块：

处理订单创建、修改、查询、取消等操作。

与支付接口集成，处理支付逻辑。

数据监控模块：

实时监控食堂运营数据，如订单量、人流量、菜品评价等。

提供数据驾驶舱功能，支持数据导出和异常预警。

菜品推荐模块：

基于用户历史订单和热门菜品生成个性化推荐。

使用AI算法进行推荐逻辑处理。

AI客服模块：

提供自然语言处理功能，解答用户咨询。

3. 数据层子系统

MySQL数据库：

存储用户信息、食堂信息、菜品信息、订单信息等。

采用规范化设计，减少数据冗余，确保数据完整性。

4. 第三方服务子系统

支付服务：

集成微信支付接口，支持在线支付功能。

处理支付状态同步，确保交易信息实时更新。

地图服务：

集成百度地图API，展示食堂相对位置。

支持地址自动填充和自动定位功能。

AI服务：

提供智能推荐和AI客服功能。

使用外部AI服务提供商的API。

#### 3.2.2 数据流图（DFD）

顶层数据流图如下所示：



#### 3.3 设计原理

#### 

#### 3.3.1 设计理由

1. 分层架构：

理由：分层架构能够清晰地划分系统的职责，提高系统的可维护性和可扩展性。前端负责用户交互，后端处理业务逻辑，数据层负责数据存储，第三方服务提供外部功能支持。

优点：各层之间通过接口交互，降低了模块间的耦合度，便于独立开发和测试。

2. 微服务架构：

理由：后端采用Springboot微服务架构，能够更好地应对系统的高并发需求，支持灵活的扩展和部署。

优点：微服务之间通过RESTful API通信，每个服务可以独立开发、部署和扩展，提高了系统的灵活性和可维护性。

3. 数据库选择：

MySQL：作为关系型数据库，MySQL能够满足系统的数据存储需求，支持复杂的查询和事务处理。

4. 第三方服务集成：

理由：通过集成第三方服务（如微信支付、地图API、AI服务等），能够快速实现系统的功能需求，减少开发成本和时间。

优点：利用成熟的第三方服务，确保系统的稳定性和可靠性。

1. 数据设计

#### 4.1数据说明

#### 4.1.1信息域到数据结构的转换

|  |  |  |
| --- | --- | --- |
| 信息域 | 数据结构 | 转换规则 |
| 用户信息 | users表（主键：user\_id） | 微信OpenID映射为用户唯一标识，加密存储敏感信息（如手机号） |
| 菜品数据 | dishes表（关联windows表） + Redis缓存热门菜品 | 菜品分类通过category字段（如“川菜”“面食”）标记，库存实时更新。 |
| 订单信息 | orders表（主键：order\_id） + order\_details子表 | 订单状态（status字段）包括：0-待支付、1-已接单、2-已完成、3-已取消。 |
| 评价数据 | reviews表（关联dishes和users） | 评分（rating字段1-5星）与文字评价分开存储，支持图片URL存储。 |
| 人流量数据 | Redis哈希表（键：canteen:flow:{campus}:{floor}） | 每5分钟更新一次，存储时间戳与人数（如`{"time": "12:00", "count": 150}`） |

#### 4.1.2主要数据实体的存储与处理

1. 用户实体（users表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| User\_id | VARCHAR(32) | 主键，微信OpenID加密哈希值 |
| nickname | VARCHAR(50) | 微信昵称 |
| favorite\_ids | JSON | 收藏的菜品ID数组（如`["D001", "D003"]`） |

处理逻辑：

注册：通过微信授权获取`nickname`和`OpenID`，自动生成`user\_id`。

更新：敏感信息（如`phone`）加密后存储，解密需密钥权限。

1. 订单实体（order表)

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| Order\_id | VARCHAR(20) | 格式：YYYYMMDD+6位随机数（如20240315012345） |
| User\_id | VARCHAR(32) | 关联users表，标识下单用户 |
| Window\_id | VARCHAR(10) | 关联windows表，标识取餐窗口 |
| Total\_amount | DECIMAL(10,2) | 订单总金额（单位：元），计算规则：SUM(order\_details.price \* quantity)。 |
| Status | TINYINT | 状态枚举：0-待支付，1-已接单（厨房准备中），2-已完成，3-已取消 |
| Pickup\_time | TIMESTAMP | 用户预约的取餐时间（若为堂食订单） |

#### 4.1.3数据库及数据存储项清单

1.MySQL数据库表

|  |  |  |
| --- | --- | --- |
| 表名 | 主要字段 | 用途 |
| Users | user\_id, nickname, phone | 用户基础信息管理 |
| windows | window\_id, canteen\_id, name | 食堂窗口信息 |
| dishes | dish\_id, window\_id, name, price, stock | 菜品信息与库存管理 |
| orders | order\_id, user\_id, status, total\_amount | 订单主记录 |
| Order\_details | detail\_id, order\_id, dish\_id, quantity | 订单明细（菜品与数量） |
| reviews | review\_id, user\_id, dish\_id, rating | 菜品评价与评分 |

2.Redis缓存项

|  |  |  |
| --- | --- | --- |
| 键格式 | 数据类型 | 说明 |
| dish:popular:{canteen\_id} | ZSET | 热门菜品排行榜（按销量排序） |
| user:session:{user\_id} | STRING | 用户会话临时数据（如未支付订单ID） |
| canteen:flow:{campus}:{floor} | HASH | 实时人流量数据（时间戳-人数键值对） |

#### 4.2 数据字典

1.实体/数据

|  |  |  |
| --- | --- | --- |
| 实体/数据名称 | 类型 | 描述 |
| canteen:flow:{校区}:{楼层} | Redis HASH | 存储实时人流量数据，键为时间戳（HH:MM），值为当前人数。 |
| dish:popular:{canteen\_id} | Redis ZSET | 热门菜品排行榜，按销量排序，成员为dish\_id，分数为销量。 |
| dishes | MySQL | 菜品信息，包含名称、价格、库存、所属窗口等字段 |
| order\_details | MySQL | 订单明细，记录每个订单中的具体菜品及数量。 |
| orders | MySQL | 订单主表，记录订单状态、总金额、用户关联等核心信息。 |
| reviews | MySQL | 用户对菜品的评价，包含评分、文字评价、图片链接等。 |
| user:session:{user\_id} | Redis STRING | 临时存储用户会话数据（如未支付订单ID），设置过期时间。 |
| users | MySQL | 用户账户信息，关联微信OpenID，存储昵称、加密手机号等 |
| windows | MySQL | 食堂窗口信息，包括所属食堂、楼层、营业状态等。 |

1. 功能描述
2. 用户（User）

属性：

user\_id (int): 用户唯一标识

name (string): 用户姓名

email (string): 用户邮箱

phone (string): 用户电话

password (string): 用户密码

created\_at (datetime): 注册时间

方法：

createUser(name, email, phone, password): 创建新用户

updateUser(user\_id, name, email, phone): 更新用户信息

deleteUser(user\_id): 删除用户

1. 菜品（Dish）

属性：

dish\_id (int): 菜品唯一标识

name (string): 菜品名称

description (string): 菜品描述

price (float): 菜品价格

category (string): 菜品类别

image\_url (string): 菜品图片链接

方法：

addDish(name, description, price, category, image\_url): 添加新菜品

updateDish(dish\_id, name, description, price, category, image\_url): 更新菜品信息

deleteDish(dish\_id): 删除菜品

1. 订单（Order）

属性：

order\_id (int): 订单唯一标识

user\_id (int): 下单用户ID

total\_amount (float): 订单总金额

status (string): 订单状态（如“待支付”、“已完成”等）

created\_at (datetime): 订单创建时间

方法：

createOrder(user\_id, total\_amount): 创建订单

updateOrderStatus(order\_id, status): 更新订单状态

deleteOrder(order\_id): 删除订单

1. 支付记录（Payment）

属性：

payment\_id (int): 支付记录唯一标识

order\_id (int): 关联订单ID

amount (float): 支付金额

status (string): 支付状态（如“成功”、“失败”等）

created\_at (datetime): 支付时间

方法：

processPayment(order\_id, amount): 处理支付

updatePaymentStatus(payment\_id, status): 更新支付状态

1. 食堂信息（Canteen）

属性：

canteen\_id (int): 食堂唯一标识

name (string): 食堂名称

location (string): 食堂位置

description (string): 食堂描述

image\_url (string): 食堂图片链接

方法：

addCanteen(name, location, description, image\_url): 添加食堂信息

updateCanteen(canteen\_id, name, location, description, image\_url): 更新食堂信息

deleteCanteen(canteen\_id): 删除食堂信息

1. 人流量（FootTraffic）

属性：

traffic\_id (int): 人流量记录唯一标识

canteen\_id (int): 食堂ID

count (int): 当前人流量

timestamp (datetime): 记录时间

方法：

recordFootTraffic(canteen\_id, count): 记录人流量

getFootTraffic(canteen\_id, time\_range): 获取人流量数据

1. 菜品评价（DishRating）

属性：

rating\_id (int): 评价唯一标识

dish\_id (int): 菜品ID

user\_id (int): 评价用户ID

rating (float): 评价分数

comment (string): 评价内容

created\_at (datetime): 评价时间

方法：

addRating(dish\_id, user\_id, rating, comment): 添加评价

updateRating(rating\_id, rating, comment): 更新评价

deleteRating(rating\_id): 删除评价

1. 推荐记录（Recommendation）

属性：

recommendation\_id (int): 推荐记录唯一标识

user\_id (int): 用户ID

dish\_id (int): 推荐菜品ID

timestamp (datetime): 推荐时间

方法：

generateRecommendation(user\_id): 生成推荐

recordRecommendation(user\_id, dish\_id): 记录推荐

#### 5. 组件设计

#### 5.1 前端子系统组件设计

#### 5.1.1 菜品浏览组件

* **功能描述**：展示菜品列表，支持搜索、筛选和排序功能。
* **输入/输出**：
  + 输入：用户搜索关键词、筛选条件（如价格范围、菜品类别）。
  + 输出：符合条件的菜品列表。
* **伪代码**：
* 函数 显示菜品(搜索关键词, 筛选条件):  
   如果 搜索关键词 不为空:  
   筛选后的菜品 = 根据关键词搜索菜品(搜索关键词)  
   否则:  
   筛选后的菜品 = 获取所有菜品()  
   如果 筛选条件 不为空:  
   筛选后的菜品 = 应用筛选条件(筛选后的菜品, 筛选条件)  
   返回 筛选后的菜品

#### 5.1.2 点单组件

* **功能描述**：支持用户选择菜品、添加到购物车、修改订单和提交订单。
* **输入/输出**：
  + 输入：用户选择的菜品ID、数量、备注。
  + 输出：购物车内容、订单提交结果。
* **伪代码**：
* 函数 添加到购物车(菜品ID, 数量, 备注):  
   购物车 = 获取购物车()  
   如果 菜品ID 不在 购物车 中:  
   购物车[菜品ID] = {数量: 数量, 备注: 备注}  
   否则:  
   购物车[菜品ID].数量 += 数量  
   如果 备注 不为空:  
   购物车[菜品ID].备注 = 备注  
   更新购物车(购物车)  
   返回 购物车  
    
  函数 提交订单(购物车):  
   订单 = 创建订单(购物车)  
   响应 = 向后端发送订单(订单)  
   如果 响应.成功:  
   清空购物车()  
   返回 {成功: 真, 订单ID: 响应.订单ID}  
   否则:  
   返回 {成功: 假, 错误: 响应.错误}

#### 5.1.3 支付组件

* **功能描述**：支持微信支付，处理支付流程并返回支付结果。
* **输入/输出**：
  + 输入：订单ID、支付金额。
  + 输出：支付结果（成功或失败）。
* **伪代码**：
* 函数 处理支付(订单ID, 金额):  
   支付请求 = 创建支付请求(订单ID, 金额)  
   响应 = 调用微信支付API(支付请求)  
   如果 响应.成功:  
   更新订单状态(订单ID, "已支付")  
   返回 {成功: 真, 交易ID: 响应.交易ID}  
   否则:  
   更新订单状态(订单ID, "支付失败")  
   返回 {成功: 假, 错误: 响应.错误}

#### 5.1.4 人流量查询组件

* **功能描述**：展示食堂当前人流量和历史趋势。
* **输入/输出**：
  + 输入：查询时间范围。
  + 输出：人流量数据。
* **伪代码**：
* 函数 获取人流量(时间范围):  
   人流量数据 = 从后端查询人流量(时间范围)  
   返回 人流量数据

#### 5.2 后端子系统组件设计

#### 5.2.1 订单管理组件

* **功能描述**：处理订单的创建、修改、查询和取消。
* **输入/输出**：
  + 输入：订单数据（用户ID、菜品列表、总金额等）。
  + 输出：订单状态和结果。
* **伪代码**：
* 函数 创建订单(用户ID, 菜品列表, 总金额):  
   订单ID = 生成订单ID()  
   订单 = {  
   订单ID: 订单ID,  
   用户ID: 用户ID,  
   菜品: 菜品列表,  
   总金额: 总金额,  
   状态: "待处理"  
   }  
   将订单保存到数据库(订单)  
   返回 订单ID  
    
  函数 更新订单状态(订单ID, 新状态):  
   订单 = 从数据库获取订单(订单ID)  
   如果 订单 不为空:  
   订单.状态 = 新状态  
   在数据库中更新订单(订单)  
   返回 {成功: 真}  
   否则:  
   返回 {成功: 假, 错误: "订单未找到"}

#### 5.2.2 数据监控组件

* **功能描述**：实时监控食堂运营数据，提供数据驾驶舱功能。
* **输入/输出**：
  + 输入：监控指标（订单量、人流量、菜品评价等）。
  + 输出：监控数据和异常预警。
* **伪代码**：
* 函数 监控数据():  
   订单统计 = 获取实时订单统计()  
   人流量 = 获取实时人流量()  
   菜品评价 = 获取实时菜品评价()  
   如果 订单统计.异常 或 人流量.异常 或 菜品评价.异常:  
   发送警报("检测到异常数据")  
   返回 {订单统计: 订单统计, 人流量: 人流量, 菜品评价: 菜品评价}

#### 5.2.3 菜品推荐组件

* **功能描述**：基于用户历史订单和热门菜品生成个性化推荐。
* **输入/输出**：
  + 输入：用户ID、历史订单数据。
  + 输出：推荐菜品列表。
* **伪代码**：
* 函数 推荐菜品(用户ID):  
   历史订单 = 获取用户订单历史(用户ID)  
   热门菜品 = 获取热门菜品()  
   推荐菜品 = 应用AI筛选(历史订单, 热门菜品)  
   返回 推荐菜品

#### 5.2.4 AI客服组件

* **功能描述**：提供自然语言处理功能，解答用户咨询。
* **输入/输出**：
  + 输入：用户问题。
  + 输出：回答内容。
* **伪代码**：
* 函数 回答问题(问题):  
   响应 = 调用AI聊天服务(问题)  
   返回 响应

#### 5.3 数据层子系统组件设计

#### 5.3.1 MySQL数据库组件

* **功能描述**：存储用户信息、食堂信息、菜品信息、订单信息等。
* **输入/输出**：
  + 输入：CRUD操作请求。
  + 输出：操作结果。
* **伪代码**：
* 函数 创建用户(用户):  
   在用户表中插入数据 (ID, 姓名, 邮箱, 电话) 值 (用户.ID, 用户.姓名, 用户.邮箱, 用户.电话)  
   返回 {成功: 真}  
    
  函数 获取订单(订单ID):  
   从订单表中选择 \* WHERE ID = 订单ID  
   返回 结果

#### 5.4 第三方服务子系统组件设计

#### 5.4.1 支付服务组件

* **功能描述**：集成微信支付接口，处理支付逻辑。
* **输入/输出**：
  + 输入：支付请求。
  + 输出：支付结果。
* **伪代码**：
* 函数 处理支付(订单ID, 金额):  
   支付请求 = {  
   订单ID: 订单ID,  
   金额: 金额,  
   时间戳: 获取当前时间()  
   }  
   响应 = 调用微信支付API(支付请求)  
   返回 响应

#### 5.4.2 地图服务组件

* **功能描述**：集成百度地图API，展示食堂位置。
* **输入/输出**：
  + 输入：地址或坐标。
  + 输出：地图展示。
* **伪代码**：
* 函数 显示地图(位置):  
   地图数据 = 调用百度地图API(位置)  
   返回 地图数据

#### 5.4.3 AI服务组件

* **功能描述**：提供智能推荐和AI客服功能。
* **输入/输出**：
  + 输入：用户请求。
  + 输出：AI处理结果。
* **伪代码**：
* 函数 获取推荐(用户ID):
    
   请求 = {用户ID: 用户ID}
    
   响应 = 调用AI服务API(请求)

六、 人机界面设计

6.1 用户界面概述

6.1.1用户界面介绍

用户界面主要由顶部用户头像、功能列表和底部导航栏三部分组成：

用户头像位于界面顶部，点击头像可更换图片，方便用户识别。

功能列表提供了修改密码、修改个人信息、AI客服、登录、注册和退出等关键操作，用户可以轻松访问和管理自己的账户信息。

底部导航栏则包含首页、点单、购物车、订单和我的等选项，帮助用户快速导航到应用程序的不同部分。整体设计注重用户体验，确保用户能够直观、便捷地使用应用程序的各项功能。

6.1.2设计原则

简洁性：界面设计简洁，功能分类明确，用户可以快速找到所需功能。

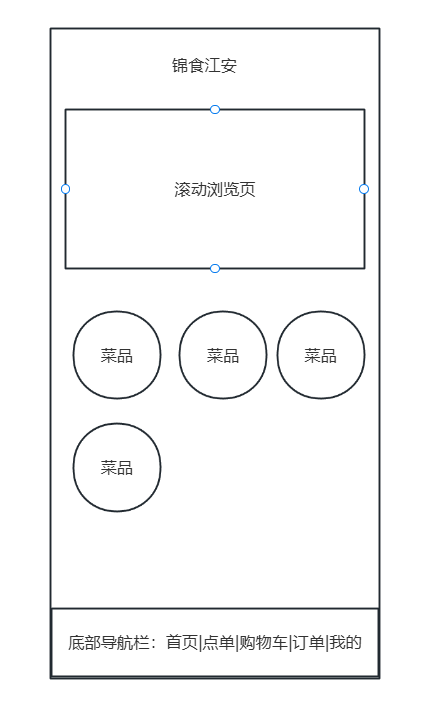
一致性：图标和文字风格一致，增强了界面的统一感。

易用性：功能按钮布局合理，用户可以轻松进行操作。

可访问性：图标和文字大小适中，确保不同用户都能轻松阅读和点击。

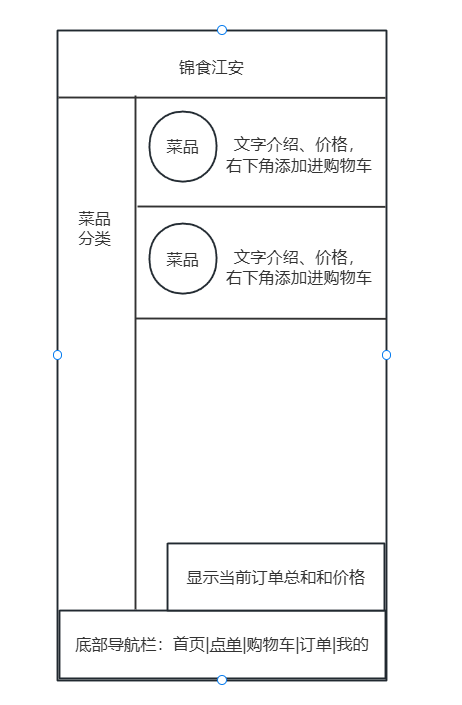
6.2 屏幕图像

6.2.1首页界面

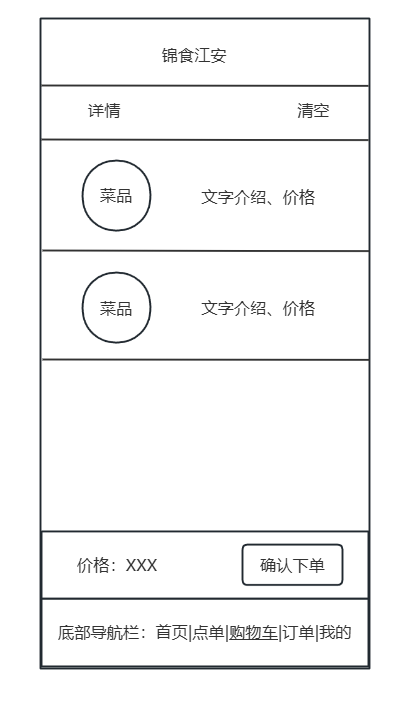


6.2.2交易界面

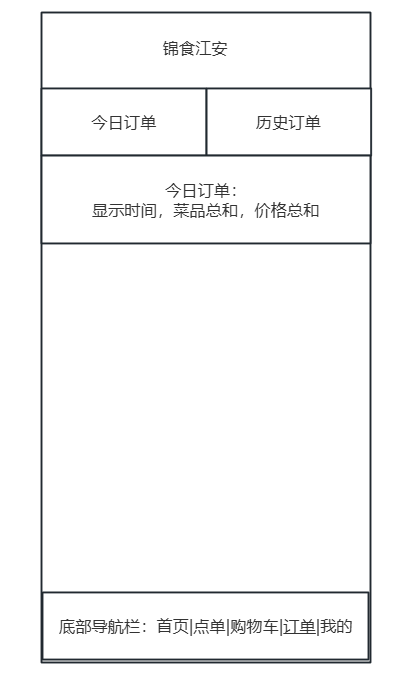
1. 点单界面



1. 购物车界面



1. 订单界面



6.2.3 个人用户界面



6.3 屏幕对象和动作

在“锦食江安”微信小程序的人机交互设计中，屏幕对象与动作的设计遵循了直观性与操作便捷性原则。

**首页界面**采用分层式信息架构，顶部区域设置品牌标识与视觉焦点元素（横幅图片）形成第一视觉层级；核心功能区域“为你推荐”采用网格布局展示菜品缩略图与价格标签，每个菜品卡片均绑定点击事件触发详情查看或加入购物车流程；右上角设置动态刷新控件（“换一换”按钮）通过点击事件实现推荐列表的异步更新。底部采用固定定位的导航栏组件，包含五个标准化图标按钮，通过触摸事件实现主要功能模块间的跳转，其选中状态通过视觉反馈（如颜色变化）明确标识当前所在页面。

**分类菜单界面**采用左右分栏式布局，左侧垂直列表呈现餐次分类标签（早餐/午餐/晚餐/夜宵），通过点击事件触发右侧内容区的动态筛选；右侧菜品展示区采用瀑布流布局，每个菜品单元包含图像、文本描述及价格信息，点击动作触发与首页相同的商品选择流程。该界面保持底部导航栏的全局一致性，确保用户在任何分类下均可快速切换功能模块。

**购物车界面**采用状态敏感型设计，顶部操作栏集成详情查看与清空功能按钮，通过确认弹窗防止误操作；内容区动态显示当前商品列表或空状态提示（含引导性操作按钮“去点单”）；底部结算栏固定显示实时计算的总金额与显性化操作按钮（“确认下单”），点击后触发订单创建流程。该界面通过视觉层次区分商品信息区与操作区，确保核心功能触达效率。

**订单管理界面**采用标签页组件实现时间维度分类（今日订单/历史订单），通过横向滑动或点击切换视图；内容区根据数据状态动态显示订单卡片列表或缺省提示（含指向性操作引导），每个订单卡片包含基础信息概要并绑定点击事件以展开详情。界面维持底部导航栏的全局控制功能，符合用户心智模型。

**个人中心界面**采用用户画像优先的布局策略，顶部头像区域绑定点击事件跳转至个人信息编辑页；功能入口采用图标+文本的列表形式排列，包含账户管理（修改密码、个人信息）、辅助功能（AI客服交流）及身份验证（登录注册/退出）三类操作，每个条目均设置标准点击响应。该界面延续底部导航模式，保持整体交互范式的一致性。所有界面均遵循微信小程序设计规范，操作反馈时间控制在500ms以内，关键路径操作设置加载状态指示，确保用户操作的可见性与系统响应可预期性。

# 七、需求跟踪矩阵

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **用户需求跟踪编号** | **用户需求标题** | **用户需求变更标识** | **软件需求功能标题** | **软件需求变更标识** | **需求状态** | **变更序号** | **优先级** | **优先级说明** | **跟踪状态** | **重要设计** |
| UD1 | 浏览食堂窗口 | 原始 | 分校区、分楼层展示食堂窗口 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD2 | 菜品搜索与收藏 | 原始 | 支持菜品搜索与收藏功能 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD3 | 在线下单 | 原始 | 在线下单（堂食/自提） | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD4 | 多方式支付 | 原始 | 支持多方式支付（微信/校园卡） | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD5 | 查看人流量热力图 | 原始 | 查看食堂人流量热力图 | 原始 | 已完成 |  | 中 | 重要功能，推荐实现 | 已实现 | 推荐实现 |
| UD6 | 菜品库存预警 | 原始 | 查看菜品库存预警 | 原始 | 已完成 |  | 中 | 重要功能，推荐实现 | 已实现 | 推荐实现 |
| UD7 | 窗口评分排行榜 | 原始 | 查看窗口评分排行榜 | 原始 | 已完成 |  | 中 | 重要功能，推荐实现 | 已实现 | 推荐实现 |
| UD8 | 发布带图评价 | 原始 | 发布带图评价 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD9 | 参与新品投票 | 原始 | 参与新品投票 | 原始 | 已完成 |  | 中 | 重要功能，推荐实现 | 已实现 | 推荐实现 |
| UD10 | 领取优惠券 | 原始 | 领取优惠券 | 原始 | 已完成 |  | 中 | 重要功能，推荐实现 | 已实现 | 推荐实现 |
| UD11 | AI客服对话 | 原始 | 与学生进行流畅对话，解答常见问题 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD12 | 实时监控订单量 | 原始 | 实时监控订单量 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD13 | 调整窗口营业状态 | 原始 | 调整窗口营业状态 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD14 | 更新菜品信息 | 原始 | 更新菜品信息 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |
| UD15 | 推送促销活动 | 原始 | 推送促销活动 | 原始 | 已完成 |  | 中 | 重要功能，推荐实现 | 已实现 | 推荐实现 |
| UD16 | 自动归类用户评价 | 原始 | 自动归类用户评价，生成改进建议报告 | 原始 | 已完成 |  | 高 | 关键功能，必须实现 | 已实现 | 必须实现 |