食堂点单小程序系统测试计划

**锦食江安**

**1. 概述**

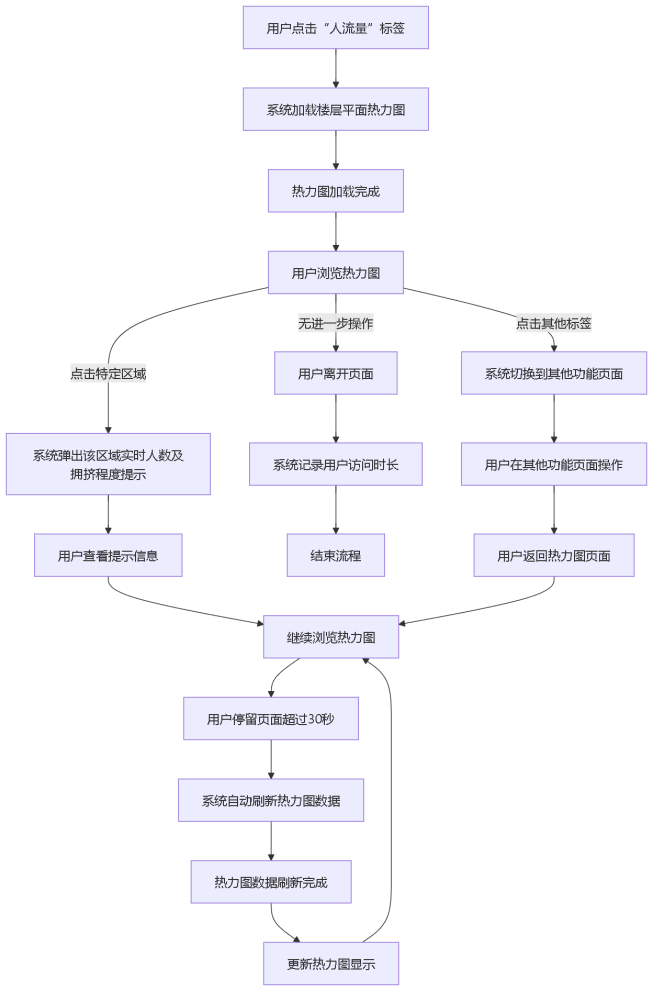
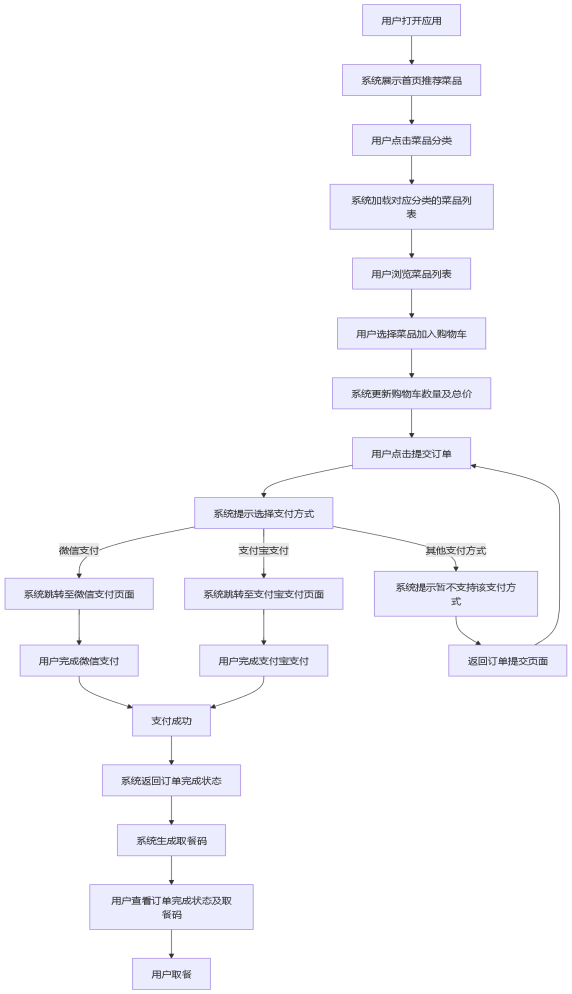
**1.1 系统需求概述**

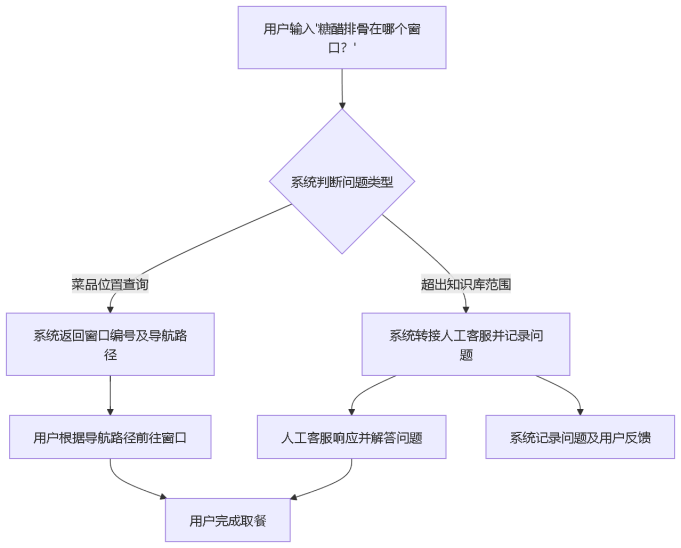
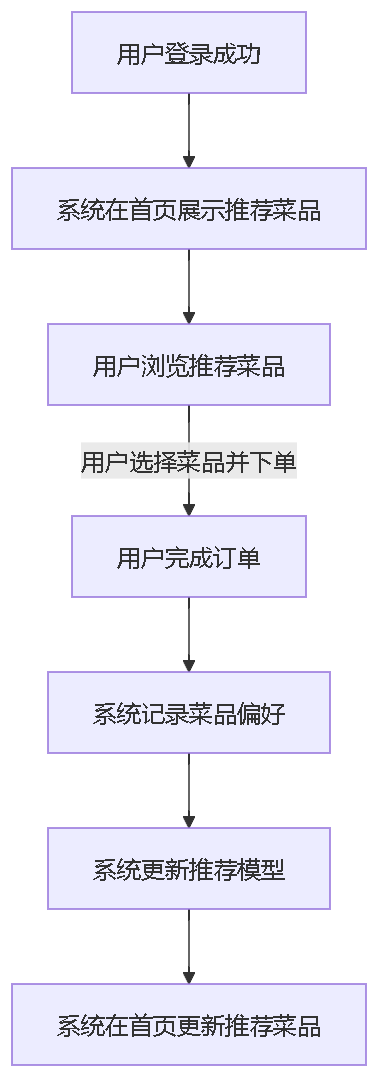
本系统是一款基于微信平台开发的轻量化点餐应用，服务对象主要为高校及企事业单位的在校师生和员工，旨在优化就餐流程、提升服务体验。用户可通过该系统实现从菜品浏览到支付完成的全流程操作，管理端则负责运营数据的监控与管理。

系统需求采用结构化编号方式组织，功能覆盖从普通用户的基本操作到后台管理的辅助功能。主要需求包括：

* 在线下单与支付
* 实时人流量查询
* 智能推荐系统
* AI客服系统

|  |  |  |  |
| --- | --- | --- | --- |
| **系统需求标题** | **软件需求功能编号** | **软件功能需求描述** | **优先级** |
| 浏览食堂窗口菜品 | REQ-01 | 展示推荐菜品和菜品分类 | 高 |
| 在线下单与购物车管理 | REQ-02 | 菜品加入购物车并提交订单 | 高 |
| 多方式支付 | REQ-03 | 微信/支付宝/其他支付方式集成 | 高 |
| 取餐服务 | REQ-04 | 订单状态展示和取餐码生成 | 高 |
| 查看人流量热力图 | REQ-05 | 展示实时人流量、支持楼层/区域切换与更新 | 中 |
| 智能推荐菜品 | REQ-06 | 记录菜品偏好，更新推荐模型 | 中 |
| AI客服对话 | REQ-7 | 提供自然语言交流，支持方言和语音输入 | 中 |





系统功能需求通过需求跟踪矩阵进行完整标识与关联。矩阵中每条功能实现项，包含需求编号、功能描述、优先级等字段，便于后续测试用例的溯源和验证。当前矩阵已体现从用户角色到功能的完整映射关系，适用于系统测试全过程中的需求可追踪性管理。

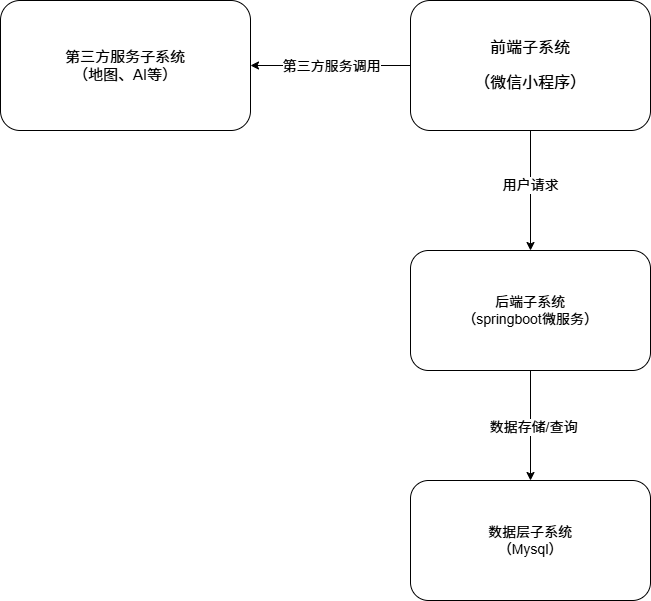
**1.2 系统设计与实现概述**

**1.2.1架构设计**

本系统采用“前后端分离 + 微服务部署”的分层架构，系统划分为四个主要子系统：

* 前端子系统：包括微信小程序用户端及后台管理Web端，负责用户交互与界面展示。
* 后端子系统：基于SpringBoot框架，承担订单处理、推荐逻辑、AI客服、数据监控等核心业务功能。
* 数据层子系统：采用MySQL管理核心业务数据，Redis实现人流量、热门菜品等缓存存储。
* 第三方服务子系统：集成微信支付、地图服务、AI接口，实现外部资源调用。

各子系统通过RESTful API进行解耦协作，整体设计强调模块独立、服务可扩展、接口标准化，保障系统性能与可维护性。



**1.2.2数据设计**

系统围绕“用户-菜品-订单-评价-人流量”构建核心数据模型：

* 用户数据通过微信OpenID唯一标识，敏感信息加密存储
* 菜品信息支持分类管理和实时库存更新，并通过缓存实现高频访问优化
* 订单数据以主子表结构组织，状态追踪涵盖待支付、已接单、已完成、取消等
* 用户评价包括评分、文字与图片字段，支持菜品满意度分析
* 人流量数据以哈希结构存储，按食堂楼层每5分钟更新一次

数据库结构设计规范化、易扩展，采用多级索引与事务控制保障查询效率与数据一致性。

|  |  |  |
| --- | --- | --- |
| 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}`） |

**1.2.3功能设计**

系统功能设计按照模块职责进行细化实现：

* 前端功能模块：包括菜品浏览、筛选搜索、购物车管理、下单支付、实时人流量查看、订单管理等，交互逻辑清晰、响应迅速。
* 后端服务模块：实现订单流转逻辑、支付状态同步、用户行为分析、AI菜品推荐与智能客服应答。
* 数据库访问模块：封装各类数据操作，确保高并发场景下的数据一致性与完整性。
* 第三方服务对接模块：对接微信支付、地图展示、AI智能推荐与客服接口，确保外部依赖服务的稳定可用。

系统整体功能满足就餐用户快速完成点餐支付的流程化需求，同时支持管理人员进行食堂运营优化、数据分析与用户反馈采集等。

**2. 测试计划**

**2.1 测试环境**

微信开发者工具：稳定版 Stable Build (1.06.2503300)

调试基础库版本：3.8.3

测试服务器：本地部署服务器

测试小程序账号：user

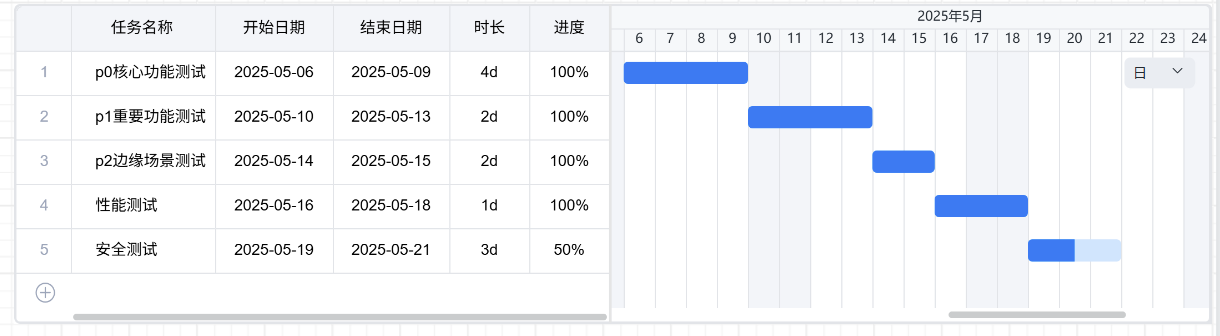
真机测试：Android 8.0.57以上版本 ios8.0.57以上版本

**2.2 测试需求IADT分解**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Function | Input | Action | Data | Test |
| P0  (核心功能) | 用户登录功能 | 用户名/密码组合 | 认证流程 | 加密存储验证 | 并发登录测试 |
| 浏览食堂窗口 | 无 | 浏览流程 | 访问窗口数据 | 并发浏览测试 |
| 菜品搜索与收藏 | 菜品名称 | 搜索流程 | 访问搜索菜品 | 特殊符号测试 |
| 在线下单 | 购物车、地址信息 | 下单流程 | 提交下单数据 | 空订单测试 |
| 发布带图评价 | 文字评价/图片 | 评价流程 | 提交评价数据 | 空评价测试 |
| P1  (重要功能) | 调整窗口营业状态 | 窗口状态更改 | 开/歇业流程 | 状态数据更新 | 状态刷新测试 |
| 更新菜品信息 | 菜品信息 | 更新菜品流程 | 菜品数据更新 | 信息更新测试 |
| 多方式支付 | 支付方式/密码 | 支付流程 | 加密支付到账 | 余额不足测试 |
| 窗口评分排行榜 | 无 | 评分查看流程 | 访问评分数据 | 并列分数测试 |
| 查看人流量热力图 | 无 | 人流查看流程 | 访问人流量 | 图片查看测试 |
| 菜品库存预警 | 菜品名称 | 菜品库存查看 | 访问菜品库存 | 库存更新测试 |
| 实时监控订单量 | 无 | 订单量查看 | 订单数据 | 订单更新测试 |
| P2  (边缘场景) | 参与新品投票 | 投票信息 | 投票流程 | 更新投票数据 | 并发投票测试 |
| 领取优惠券 | 优惠券信息 | 领取流程 | 更新券数据 | 领取确认测试 |
| AI客服对话 | 对话 | 对话回应 | AI接入传递 | 多并发对话 |
| 推送促销活动 | 无 | 推送流程 | 访问菜品数据 | 活动更新测试 |
| 自动归类用户评价 | 用户评价 | 归类流程 | 评价数据排列 | 分类查看测试 |

**2.3 测试实施**

1.系统验收测试甘特图

  
2.验证和需求测试文档的符合：

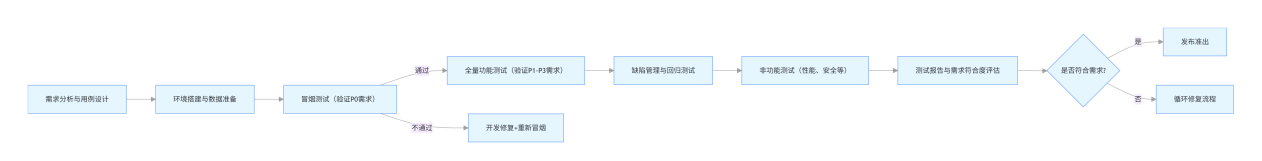
****需求文档与测试文档的一致性验证，目标为**确保测试覆盖所有需求点，且测试逻辑与需求描述一致。**

1. 测试人员与分工：

| ****角色**** | ****职责描述**** | ****需求验证重点**** | ****负责人**** |
| --- | --- | --- | --- |
| ****测试负责人**** | 制定测试计划，协调资源，审核测试文档与需求的一致性。 | 确保测试策略覆盖需求优先级（如 P0 需求优先测试） | 唐纳 |
| ****功能测试工程师**** | 编写功能测试用例，执行冒烟测试、回归测试，验证需求实现细节。 | 字段校验、流程跳转、业务规则正确性 | 唐纳、刘小瑜、唐雨涵 |
| ****性能测试工程师**** | 设计性能测试方案，模拟负载场景，验证响应时间、TPS 等非功能需求。 | 需求中的性能指标是否达标 | 刘小瑜 |
| ****安全测试工程师**** | 执行 OWASP Top10 漏洞扫描、渗透测试，确保需求中的安全要求（如数据加密）落地。 | 验证加密算法、权限控制是否符合需求规格 | 唐雨涵 |

4.测试方式：

| ****测试方式**** | ****定义**** | ****需求验证重点**** | ****工具 / 方法**** |
| --- | --- | --- | --- |
| ****冒烟测试**** | 验证核心需求是否可基本运行（如 P0 需求）。 | 主流程畅通性（如登录→下单→支付） | 手工执行 + 自动化脚本（如 Jenkins） |
| ****功能测试**** | 基于需求文档，验证功能点的正确性、完整性。 | 分支流程、异常场景（如订单金额为 0 的提示） | 手工用例 + Postman 接口测试 |
| ****性能测试**** | 验证非功能需求（如响应时间、吞吐量、稳定性）。 | 需求中的性能指标（如日均处理订单量≥10 万） | JMeter、Gatling |
| ****安全测试**** | 检测需求中的安全要求（如身份验证、数据加密、防攻击）。 | 漏洞扫描（如 SQL 注入、XSS）、权限验证 | Burp Suite、OWASP ZAP |
| ****兼容性测试**** | 验证需求在不同环境下的一致性（如浏览器、设备、操作系统）。 | 界面适配、功能一致性（如移动端订单页显示） | BrowserStack、模拟器 |

1. 测试如何具体进行  
   

**2.4 通过准则**  
量化标准如下。  
**2.4.1功能测试**

100% P0用例通过且P1通过率≥95%

功能测试的基础质量门槛，其核心价值在于：

保障用户基本体验：确保用户能完成核心操作，避免 “不可用” 事故。

控制上线风险：通过分级管理，平衡质量与效率，允许非关键缺陷通过后续迭代优化。

推动团队质量意识：将 “核心功能零缺陷” 作为硬性标准，倒逼开发与测试团队提升需求分析、用例设计和代码质量。

1. 100% P0 用例通过

在测试计划中，P0 用例需作为首轮测试项，确保核心功能提前验证。若P0用例失败，需立即触发阻塞流程，开发团队需优先修复（如暂停其他任务，集中解决P0缺陷）。

1. P1通过率≥95%

允许P1用例存在“≤5% 的失败率”，但需满足以下条件：

（1）失败用例不涉及数据安全、用户隐私、法规合规等高危风险。

（2）失败场景需评估对用户体验的影响程度（如偶发的界面显示异常可接受，但频繁的功能报错不可接受）。

允许少量非关键 P1 用例遗留，但需明确记录缺陷，并制定闭环跟踪计划（如版本发布后 X 个工作日内修复）。

**2.4.2性能测试**

TPS波动≤5%，错误率<0.1%

性能测试的稳定性量化指标，其核心目标是：保障业务连续性：避免因 TPS 剧烈波动导致用户体验忽好忽坏（如页面加载速度时快时慢）；降低故障风险：通过严格控制错误率，防止偶发缺陷积累成系统性故障（如 1% 的错误率在千万级请求中会导致 10 万次失败）；

支撑架构优化：以指标为导向，推动技术团队从代码逻辑、缓存策略、硬件配置等多维度提升系统韧性。

1. TPS波动≤5%

TPS（Transactions Per Second），每秒处理的事务数，是衡量系统处理能力的核心指标。  
 TPS波动≤5%，指在稳定负载阶段，TPS 的实时值与平均值的偏差需控制在 ±5% 以内。意义在于，波动过大可能反映系统存在资源竞争（如锁冲突）、缓存失效或架构瓶颈（如数据库连接池不足）。

2. 错误率 < 0.1%

测试过程中，失败事务数占总事务数的比例需低于 0.001（即每 1000 次请求中失败≤1 次）。失败类型包括接口返回错误码（如 HTTP 500）、超时、数据不一致等。

**2.4.3安全测试**

OWASP Top10漏洞零发现

OWASP（Open Web Application Security Project）是全球权威的 Web 应用安全标准，其发布的《OWASP Top10》是年度更新的十大最严重 Web 漏洞列表，覆盖注入、跨站脚本（XSS）、身份验证失效、敏感数据暴露等核心风险。OWASP Top10 漏洞覆盖约 80% 的 Web 攻击场景，消除这些漏洞可大幅降低数据泄露、服务瘫痪等核心风险。

|  |  |  |
| --- | --- | --- |
| 序号 | 漏洞类型 | 典型场景 |
| 1 | ****注入**** | SQL/OS 命令注入，如用户输入未过滤导致数据库信息泄露 |
| 2 | ****身份验证失效**** | 弱密码、会话固定、令牌窃取等导致用户身份被冒用 |
| 3 | ****跨站脚本**** | 恶意脚本嵌入网页，窃取用户 Cookie 或执行钓鱼攻击 |
| 4 | ****XML 外部实体**** | 利用 XML 解析器加载外部实体，导致服务器文件读取或端口扫描 |
| 5 | ****访问控制失效**** | 越权访问（如普通用户绕过权限查看管理员页面） |
| 6 | ****安全配置错误**** | 未关闭调试模式、默认账号密码、文件权限配置错误等 |
| 7 | ****跨站请求伪造**** | 伪造用户请求执行非法操作（如转账、修改密码） |
| 8 | ****不安全的反序列化**** | 反序列化过程中植入恶意代码，导致远程代码执行（RCE） |
| 9 | ****使用过时的组件**** | 引用存在已知漏洞的第三方库（如 Log4j 2.x 早期版本） |
| 10 | ****日志记录与监控不足**** | 缺乏攻击日志记录，导致无法及时发现漏洞利用或数据泄露 |

****2021 版 OWASP Top10 漏洞列表****（当前广泛应用的版本）

**2.4.4出现测试问题的等级**

1. 缺陷等级定义与分类标准

通常按影响程度和紧急程度将缺陷分为 4-5 级，以下为通用分级模型：

| 等级 | 名称 | 定义 | 典型场景 |
| --- | --- | --- | --- |
| 1级 | 致命缺陷 | 导致系统崩溃、数据丢失或严重安全漏洞，直接阻断业务流程。 | 支付失败且无事务回滚、用户密码明文存储、SQL 注入漏洞。 |
| 2级 | 严重缺陷 | 影响核心功能使用，但系统仍可运行（需用户绕行或存在明显体验问题）。 | 订单状态显示错误、搜索功能无结果返回、文件上传失败但无错误提示。 |
| 3级 | 一般缺陷 | 非核心功能异常或体验瑕疵，不影响主要业务流程。 | 界面元素错位、提示信息不准确、操作流程冗余（如多步完成可简化为一步）。 |
| 4级 | 轻微缺陷 | 视觉 / 交互细节问题，对用户影响极小（可在下一迭代优化）。 | 按钮颜色不符合设计规范、文案错别字、加载动画不流畅。 |
| 5级 | 建议项 | 非缺陷，属于优化建议（如性能提升、易用性改进）。 | 建议增加批量操作功能、优化报表加载速度。 |

2. 各等级缺陷的处理优先级

| 等级 | 修复时限 | 发布影响 | 评审要求 |
| --- | --- | --- | --- |
| 1级 | 立即修复（24 小时内） | 禁止带缺陷上线，需验证通过后才可发布。 | 需开发、测试、产品经理三方紧急评审。 |
| 2级 | 版本发布前修复 | 高风险 2 级缺陷禁止遗留，中低风险可协商。 | 需测试负责人与产品经理确认风险。 |
| 3级 | 按迭代计划处理 | 允许遗留，但需在测试报告中说明影响。 | 低风险缺陷可直接列入待办列表。 |
| 4级/ 5级 | 非强制修复 | 不影响发布，可作为技术债记录。 | 由开发团队自主决定是 |

**3. 测试程序**

**3.1 测试程序组成**

测试组成：

测试比例：功能测试：60% | 性能测试：20% 接口测试：15% | 其他：5%

**3.2 功能测试**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **需求规格** | **操作步骤** | **期望输出** | **实际输出** | **通过与否** | **优先级** |
| 1 | 首页菜品展示 | 1. 进入小程序首页 | 正常加载所有图片，无空白/错位 |  |  | P0 |
| 2. 查看顶部轮播图和固定菜品区 |
| 2 | 添加商品到购物车 | 1. 点击任意菜品 | 购物车角标数量+1，底部显示总价 |  |  | P0 |
| 2. 选择配菜/配送方式 |
| 3. 点击"加入购物车" |
| 3 | 订单生成与支付 | 1. 购物车添加商品 | 生成待配送订单，状态为"已支付" |  |  | P0 |
| 2. 点击"结算" |
| 3. 完成支付流程 |
| 4 | 用户登录 | 1. 点击"我的" | 跳转到个人中心页，显示用户信息 |  |  | P0 |
| 2. 输入正确账号密码 |
| 3. 点击登录 |
| 5 | 餐段切换（早/中/晚） | 1. 进入菜品页 | 仅显示早餐类商品，列表更新 |  |  | P1 |
| 2. 点击左侧"早餐"标签 |
| 6 | 配送方式选择 | 1. 添加商品时选择"打包" | 订单详情中显示"打包" |  |  | P1 |
| 2. 查看结算页 |
| 7 | 订单状态查询 | 1. 支付完成后 | 订单显示取餐号，含预计取餐时间 |  |  | P1 |
| 2. 进入"订单"页查看 |
| 8 | 定位功能 | 1. 点击"我的"页定位图标 | 显示当前位置据食堂距离 |  |  | P1 |
| 9 | "换一换"推荐 | 1. 首页点击"换一换"按钮 | 固定菜品区内容更新，无重复项 |  |  | P2 |
| 10 | 注册新账号 | 1. 点击"注册" | 跳转到登录页，提示"注册成功" |  |  | P2 |
| 2. 填写合规信息 |
| 3. 提交 |
| 11 | 退出登录 | 1. 已登录状态下 | 返回登录页，清除用户数据 |  |  | P2 |
| 2. 点击"退出" |

**3.3 性能测试**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **需求规格** | **操作步骤** | **期望输出** | **实际输出** | **通过与否** |
| 1 | 首页加载性能 | 1. 冷启动小程序 | 加载时间≤1.5s，图片完整渲染 |  |  |
| 2. 进入首页 |
| 2 | 菜品列表加载 | 1. 切换早餐/午餐/晚餐分类 | 列表加载时间≤1s，无卡顿 |  |  |
| 3 | 购物车结算并发 | 模拟50用户同时提交订单 | 成功率100%，平均响应时间≤2s |  |  |
| 4 | 订单列表刷新 | 1. 下拉刷新历史订单 | 新数据加载时间≤800ms |  |  |
| 5 | 登录响应 | 1. 输入正确账号密码提交 | 跳转延迟≤1s |  |  |

**3.4 接口测试**

| **序号** | **接口名称** | **测试步骤** | **期望输出** | **实际输出** | **通过与否** |
| --- | --- | --- | --- | --- | --- |
| 1 | 菜品分类接口 | 1. 请求/api/category | 返回早餐分类及对应菜品ID/名称 |  |  |
| 2. 传参type=breakfast |
| 2 | 购物车添加接口 | 1. POST/api/cart/add | 返回更新后的购物车总价及商品数量 |  |  |
| 2. 传参菜品ID+配菜选项 |
| 3 | 订单提交接口 | 1. POST/api/order/submit | 返回订单编号及预计配送时间 |  |  |
| 2. 传参用户token+购物车数据 |
| 4 | 登录验证接口 | 1. POST/api/user/login | 正确时返回token，错误时返回401 |  |  |
| 2. 传参正确/错误密码组合 |
| 5 | 定位获取接口 | 1. GET/api/location | 返回经纬度及食堂地址文字 |  |  |
| 2. 携带用户token |

**3.5 文档介值一致性测试**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **文档名称** | **版本号** | **关键配置项** | **实际环境值** | **是否一致** | **修改记录（若不一致）** |
| 《服务器清单文档》 | V1.0.2 | 服务器 IP 地址、端口号 | 192.168.1.100:8080 |  | - |
| 《数据库设计文档》 | V2.1.0 | 数据库名、用户名、密码 | db\_app, app\_user, \*\*\*\*\*\*\*\*\* |  | - |
| 《应用配置文档》 | V3.2.1 | 日志存储路径、接口超时时间 | /data/logs, 5000ms |  |  |
| 《中间件配置文档》 | V1.1.1 | Tomcat 最大线程数 | 200 |  | - |
| 《网络拓扑文档》 | V1.0.1 | 负载均衡转发规则 | 80 端口转发至 8080 |  | - |

**3.6 自由操作测试**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **操作类型** | **测试动作** | **预期系统反应** | **实际结果** | **严重等级** |
| 1 | 快速连续点击 | 在0.5秒内连续点击"结算"按钮10次 | 仅生成1个订单，其余请求被过滤 |  | P0 |
| 2 | 异常输入 | 在数量输入框输入"abc" | 提示"请输入数字"并清空输入框 |  | P1 |
| 3 | 中断操作 | 支付过程中突然关闭小程序 | 恢复后显示未完成订单待支付 |  | P0 |
| 4 | 反向操作 | 在订单详情页连续上拉刷新20次 | 保持稳定不卡顿 |  | P1 |