|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QQ截图20130904105420  **餐馆信息管理系统**  **系统测试报告**  设计人员： 2021112925-吴平凡  2021112923-袁旭阳  2021112937-李 品  指导教师 李 威   |  |  |  | | --- | --- | --- | | **文件状态：**  **[√ ] 草稿**  **[ ] 正式发布**  **[ ] 正在修改** | **文件标识** | **1.0** | | **当前版本** | **V1.0** | | **拟 稿 人** | 吴平凡 | | **拟稿日期** | 2024/5/30 | | **审 核 人** |  | | **审核日期** |  | |

编写说明

标题：软件需求规格说明书

类别：文档

存放位置：项目文档\04、项目需求\系统测试报告.docx

编辑软件：Miscrosoft Word 2021 中文版

版本历史：

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 日期 | 备注 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

完成情况分工：

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 工作量 | 完成工作内容 |
| 2021112923 | 吴平凡 | 80% | 完成了白盒测试、黑盒测试以及性能测试，并编写了报告 |
| 2021112925 | 袁旭阳 | 10% | 报告的审核与修改 |
| 2021112937 | 李品 | 10% | 报告的审核与修改 |
|  |  |  |  |
|  |  |  |  |

**目 录**

1. 测试项目概述 4

1.1 项目业务功能介绍 4

1.2 测试需求说明 5

1.3 验证系统功能的正确性： 5

1.4 确保系统的稳定性： 5

1.5 提升用户体验： 5

1.6 测试环境及工具 5

2. 测试计划 6

3. 测试过程及用例 6

3.1 白盒测试用例 6

3.2 黑盒测试用例 13

3.3 性能测试 17

4. 缺陷分析 19

4.1 缺陷报告 19

5. 测试总结 19

5.1 白盒测试： 19

5.2 黑盒测试： 20

5.3 性能测试： 20

5.4 结论 20

# 测试项目概述

在对平凡餐馆信息管理系统的测试过程中，我们进行了全面的测试，涵盖了白盒测试、黑盒测试以及性能测试。

## 项目业务功能介绍

图示

描述已自动生成图示

描述已自动生成

图1-1 服务层流程图示

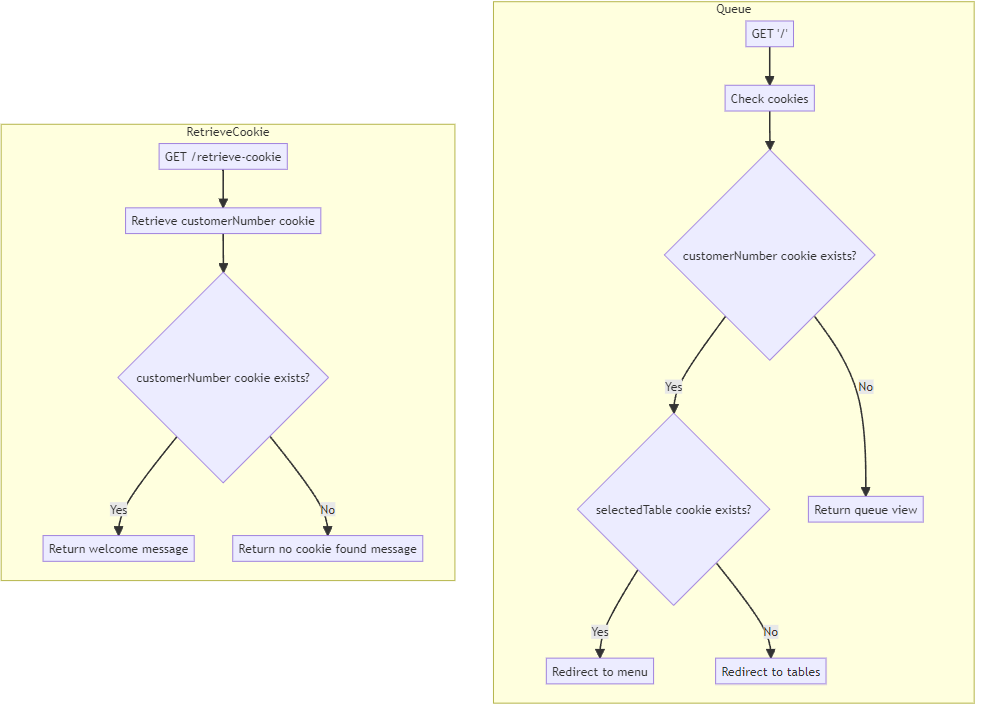


图1-2 控制层流程图示1

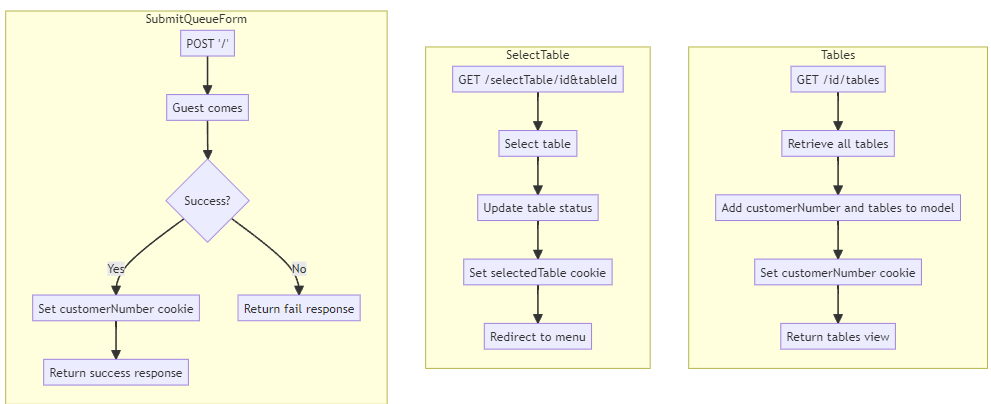


图1-2 控制层流程图示2

## 测试需求说明

本文档是根据“餐馆信息管理系统”需求分析说明书编写的测试需求说明书。其目的是确保系统功能的正确性、稳定性和用户体验。具体目标如下：

## 验证系统功能的正确性：

* + 1. 确保所有业务功能按照需求分析说明书中的描述正确实现。
    2. 验证前端和后端交互的正确性，包括数据的传输和处理。

## 确保系统的稳定性：

* + 1. 通过各种测试场景和边界条件，确保系统在不同情况下都能稳定运行。
    2. 检查系统在高负载和压力条件下的表现。

## 提升用户体验：

* + 1. 确保用户界面友好，提示信息明确，操作流程顺畅。
    2. 验证错误提示和成功页面的显示是否符合预期。

## 测试环境及工具

* + 1. 网络环境：测试环境采用本机网络。
    2. 支持系统软件：操作系统支持Windows 10/11;云数据库：MySQL8.0；应用服务器：Tomcat
    3. 应用软件：前端使用HTML5, CSS3, JavaScript；后端使用SpringBoot架构，使用Maven用于项目构建和依赖管理
    4. 接口：RESTful API，用于与前端和其他服务进行通信
    5. 测试数据需求：

测试用户数据：包括不同角色的用户（管理员、普通用户等）

测试订单数据：包括不同状态的订单（新订单、进行中、已完成）

餐桌数据：不同的餐桌状态（空闲、已预定、占用）

菜品数据：不同类别和状态的菜品

数据生成工具：Mockaroo用于生成模拟数据

* + 1. 测试工具：

Selenium：用于Web UI自动化测试

JUnit：用于单元测试

TestNG：用于集成测试

Mockito：用于模拟对象和服务

# 测试计划

测试开始日期：2023年5月24日

测试结束日期：2023年6月14日

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本号 | 说明 | 状态 |
| 2024年6月2日 | 1.0 | 基本完成白盒测试 | 基本完成 |
| 2024年6月7日 | 1.2 | 基本完成黑盒测试 | 基本完成 |

# 测试过程及用例

## 白盒测试用例

白盒测试1-基本路径覆盖：

（1）验证点菜服务层DishService 接口dishOrder 方法是否能够正确处理订单的创建和保存。

（2）代码结构

|  |
| --- |
| @Test  public void testDishOrder() {  doNothing().when(dishOrderDao).orderDish(anyString(), anyString());  // 调用实际方法  assertThrows(IllegalArgumentException.class,()->dishService.dishOrder("Table1", "Dish1"));  theTable table = new theTable();  table.setTableId("Table1");  Dish dish = new Dish();  dish.setId("Dish1");  when(tableRepository.findById(anyString())).thenReturn(Optional.of(table));  assertThrows(IllegalArgumentException.class,()->dishService.dishOrder("Table1", "Dish1"));  when(dishRepository.findById(anyString())).thenReturn(Optional.of(dish));  dishService.dishOrder("Table1","Dish1");  // 验证 dishOrderDao 是否被调用  verify(dishOrderDao, times(1)).orderDish("Table1", "Dish1");  } |

（3）覆盖测试方法-基本路径覆盖

图示

描述已自动生成

图2-1服务层dishOrder 流程

*E*=6

𝑁=6

𝑃=1

所以环路复杂度 𝐶𝐶=6−6+2×1=3*CC*=6−6+2×1=2

（4） 测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **输入数据** | **路径** | **预期结果** | **实际结果** | **错误原因** |
| 1 | tableId = "1", dishName = "" | A -> B -> C(Yes) -> D -> F | IllegalArgumentException thrown | IllegalArgumentException thrown | 无 |
| 2 | tableId = "", dishName = "1" | A -> B -> C(Yes) -> D -> F | IllegalArgumentException thrown | IllegalArgumentException thrown | 无 |
| 3 | tableId = "1", dishName = "1" | A -> B -> C(No) -> E -> F | dishOrderDao.orderDish(tableId, dishName) called | dishOrderDao.orderDish(tableId, dishName) called | 无 |

白盒测试2-条件覆盖：

（1）验证顾客服务层GuestServer接口guestCome方法是否能够正确处理顾客的到来

（2）代码结构

|  |
| --- |
| @Test  public void testGuestCome(){  when(guestDAO.guestCome(anyInt(), anyInt())).thenReturn(1);  //条件覆盖  assertThrows(IllegalArgumentException.class,()->guestService.guestCome(-1,1));  assertThrows(IllegalArgumentException.class,()->guestService.guestCome(1,-1));  // 调用被测试的方法  int result = guestService.guestCome(1,1);  // 验证 dishOrderDao.orderDish() 方法被调用  verify(guestDAO, times(1)).guestCome(1,1);  assertEquals(1,result);  } |

（3）覆盖测试方法-基本路径覆盖

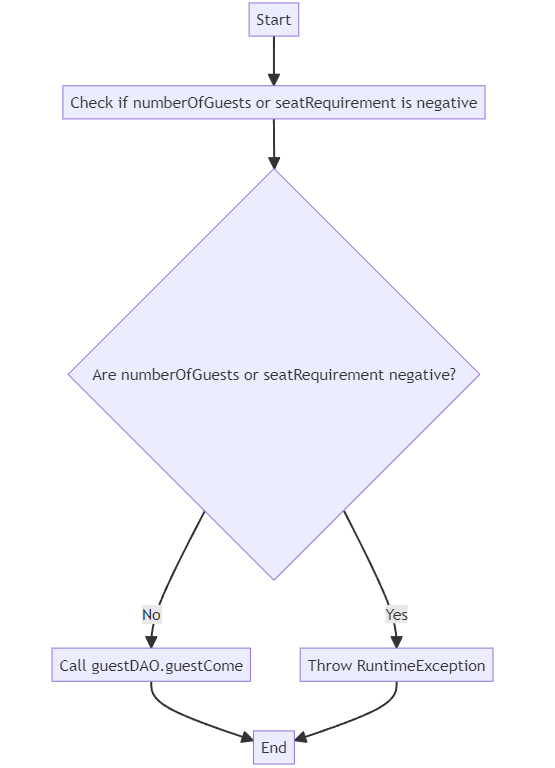


图2-2服务层guestCome流程

*E*=6

𝑁=6

𝑃=1

所以环路复杂度 𝐶𝐶=6−6+2×1=3*CC*=6−6+2×1=2

（4） 测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **输入数据** | **条件1 (numberOfGuests < 0)** | **条件2 (seatRequirement < 0)** | **预期结果** | **错误原因** |
| 1 | numberOfGuests = -1, seatRequirement = 5 | 真 | 假 | 抛出 RuntimeException | 无 |
| 2 | numberOfGuests = 5, seatRequirement = -1 | 假 | 真 | 抛出 RuntimeException | 无 |
| 3 | numberOfGuests = -1, seatRequirement = -1 | 真 | 真 | 调用 guestDAO.guestCome  (numberOfGuests, seatRequirement) 并返回结果 | 无 |

白盒测试3-语句覆盖：

（1）验证顾客服务层GuestServer接口selectTable方法是否能够正确处理顾客选择桌子的操作。

（2）代码结构

|  |
| --- |
| @Test  void testSelectTable() {  doNothing()  .when(tableRepository).guestToTable(anyInt(),anyString());  theTable table = new theTable();  table.setTableId("table1");  when(tableRepository.findById(anyString())).thenReturn(Optional.of(table));  assertThrows(IllegalArgumentException.class, ()->tableService.selectTable(-1,"table1"));  tableService.selectTable(1,"table1");  verify(tableRepository).guestToTable(1,"table1");  when(tableRepository.findById(anyString())).thenReturn(Optional.empty());  assertThrows(IllegalArgumentException.class, ()->tableService.selectTable(1,"table1"));  } |

（3）覆盖测试方法-语句覆盖

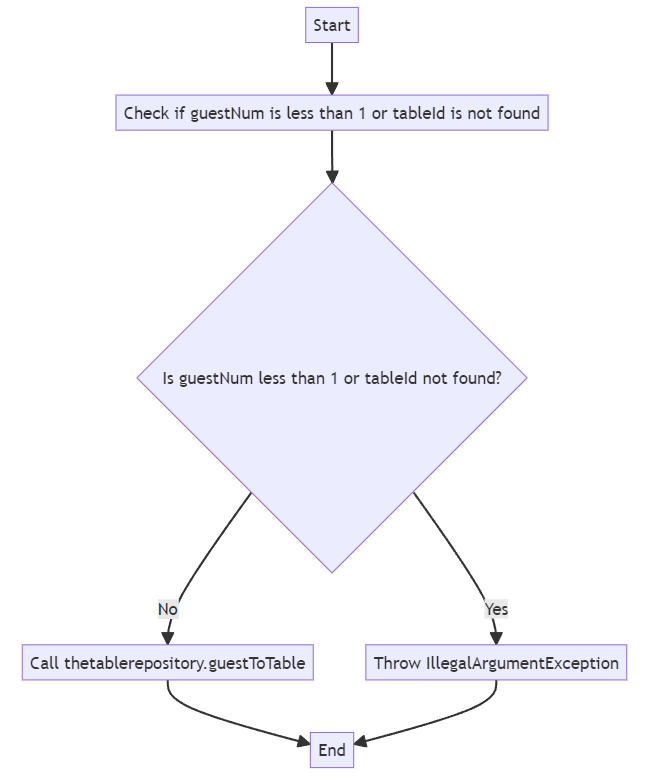


图2-3服务层selectTable流程

*E*=6

𝑁=6

𝑃=1

所以环路复杂度 𝐶𝐶=6−6+2×1=3*CC*=6−6+2×1=2

（4） 测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 输入数据 | 预期结果 | 实际结果 | 错误原因 |
| 1 | guestNum = 0, tableId = "1" | 抛出 IllegalArgumentException | 抛出 IllegalArgumentException | 无 |
| 2 | guestNum = 1, tableId = "2" | 抛出 IllegalArgumentException（假设 tableId 为 "2" 的记录不存在） | 抛出 IllegalArgumentException（假设 tableId 为 "2" 的记录不存在） | 无 |
| 3 | guestNum = 2, tableId = "3" | 调用 thetablerepository.guestToTable(guestNum, tableId) 并正常执行 | 调用 thetablerepository.guestToTable(guestNum, tableId) 并正常执行 | 无 |
| 4 | guestNum = -1, tableId = "1" | 抛出 IllegalArgumentException | 抛出 IllegalArgumentException | 无 |
| 5 | guestNum = 10, tableId = "1" | 抛出 IllegalArgumentException（假设桌子最多只能容纳 5 位客人） | 抛出 IllegalArgumentException（假设桌子最多只能容纳 5 位客人） | 无 |
| 6 | guestNum = 2, tableId = "" | 抛出 IllegalArgumentException | 抛出 IllegalArgumentException | 无 |
| 7 | guestNum = 2, tableId = null | 抛出 NullPointerException | 抛出 NullPointerException | 无 |

（5）结果分析（junit的报告）

白盒测试1-基本路径覆盖：



图2-4 白盒测试1报告

白盒测试2-条件判定覆盖：



图2-5 白盒测试2报告

白盒测试3-语句覆盖：

图2-6 白盒测试3报告

## 黑盒测试用例

黑盒测试1：获取所有桌子信息

（1）功能： 获取餐厅中的所有桌子信息并返回给前端显示。

（2）问题描述及功能界面

当用户访问特定路径（如 /{id}/tables），系统应返回所有桌子的列表，并在页面上显示顾客编号和桌子信息，同时设置一个带有顾客编号的Cookie。

（3）测试用例设计分析

1. 等价类划分法：

有效类：用户ID有效，能够获取桌子信息。

无效类：用户ID无效或不存在，无法获取桌子信息。

1. 边界值分析法：

用户ID的边界值（假设ID为正整数）：最小有效值（1）、极大值（理论上的最大用户ID）。

1. 判定表法：

判定表可以帮助我们根据用户ID的有效性、Cookie的正确性和页面重定向等条件来设计测试用例。

（4）测试用例设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试方法 | 前置条件 | 输入 | 预期结果 |
| TC1-1 | 等价类划分法 | ID有效 | 1 | 返回所有桌子信息，设置有效Cookie |
| TC1-2 | 等价类划分法 | ID无效（不存在） | 9999 | 返回错误页面或空列表，不设置Cookie |
| TC1-3 | 边界值分析法 | 最小有效ID | 1 | 返回所有桌子信息，设置有效Cookie |
| TC1-4 | 边界值分析法 | 最大有效ID | 2147483647 | 返回所有桌子信息，设置有效Cookie |
| TC1-5 | 判定表法 | ID有效，且Cookie设置正常 | 1 | 返回所有桌子信息，Cookie正确设置 |
| TC1-6 | 判定表法 | ID无效，且Cookie设置失败 | 9999 | 返回错误页面，不设置Cookie |

表2-1 黑盒测试1测试用例设计

（5）工具及脚本设计（webUI ）

使用 Selenium进行 Web UI 自动化测试，检查页面加载情况、Cookie 设置情况及重定向行为。

（6）结果/缺陷分析

所有用例都能得到正确的预期结果，说明程序设计较为完善。

黑盒测试2：选择桌子

（1）功能： 用户选择餐厅中的一个桌子，并更新桌子的状态，设置一个带有已选择桌子的Cookie，并重定向到菜单页面。

（2）问题描述及功能界面

用户在 /{id}&{tableId} 路径下选择桌子后，系统应更新桌子的状态，设置已选择桌子的Cookie，并重定向到菜单页面。

（3）测试用例设计分析

1. 因果图法：

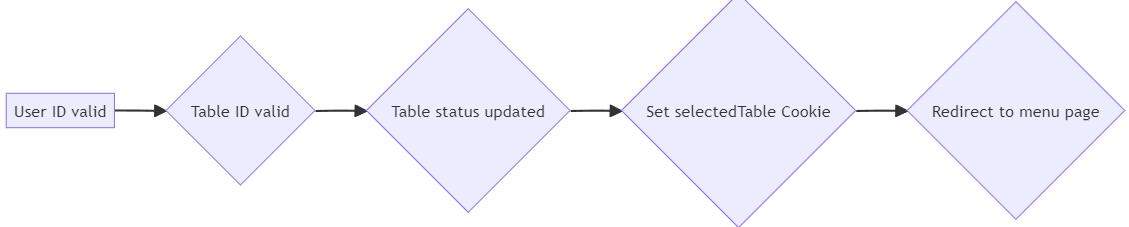
通过用户选择桌子的行为（因）导致桌子状态的更新、Cookie 的设置和页面的重定向（果）。

图2-7 因果图

1. 边界值分析法：

针对桌子ID进行边界值测试，验证最小和最大桌子ID的处理。

（4）测试用例设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试方法 | 前置条件 | 输入 | 预期结果 |
| TC2-1 | 因果图法 | 用户ID和桌子ID均有效 | /1&5 | 桌子状态更新，设置selectedTable Cookie，重定向到菜单页面 |
| TC2-2 | 因果图法 | 用户ID有效，桌子ID无效 | /1&9999 | 返回错误信息或页面，Cookie未设置 |
| TC2-3 | 边界值分析法 | 最小有效桌子ID | /1&1 | 桌子状态更新，设置selectedTable Cookie，重定向到菜单页面 |
| TC2-4 | 边界值分析法 | 最大有效桌子ID | /1&2147483647 | 桌子状态更新，设置selectedTable Cookie，重定向到菜单页面 |

表2-2 黑盒测试2测试用例设计

（5）工具及脚本设计（webUI ）

使用 Postman进行 API 调用，检查响应状态、Cookie 的设置以及页面的重定向。

（6）结果/缺陷分析

所有用例都能得到正确的预期结果，说明程序设计较为完善。

黑盒测试3：提交排队表单

（1）功能： 用户提交排队表单后，系统为用户分配一个顾客编号并设置相应的Cookie。

（2）问题描述及功能界面

用户在排队页面提交表单（POST 请求到 /），系统应处理用户的排队请求，返回一个顾客编号并设置一个带有顾客编号的Cookie。

（3）测试用例设计分析

1. 场景法：

测试不同的用户提交排队请求的场景，包括不同的顾客容量、重复提交等。

1. 正交实验法：

使用正交实验法结合不同的输入参数（如顾客容量、是否有已有顾客编号等）进行组合测试。

（4）测试用例设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试方法 | 前置条件 | 输入 | 预期结果 |
| TC3-1 | 场景法 | 有效顾客容量 | POST / {capacity: 4} | 分配顾客编号，设置customerNumber Cookie，返回成功信息 |
| TC3-2 | 场景法 | 无效顾客容量（负数） | POST / {capacity: -1} | 返回错误信息，不设置Cookie |
| TC3-3 | 场景法 | 重复提交表单 | POST / {capacity: 4} x 2 | 分配顾客编号，设置新的customerNumber Cookie，返回成功信息 |
| TC3-4 | 正交实验法 | 有效顾客容量，已有顾客编号 | POST / {capacity: 6} | 分配顾客编号，覆盖旧的customerNumber Cookie，返回成功信息 |
| TC3-5 | 正交实验法 | 有效顾客容量，无已有顾客编号 | POST / {capacity: 6} | 分配顾客编号，设置customerNumber Cookie，返回成功信息 |

表2-3 黑盒测试3测试用例设计

（5）工具及脚本设计（webUI ）

使用 Postman进行 API 调用，检查响应状态、Cookie 的设置以及页面的重定向。

（6）结果/缺陷分析

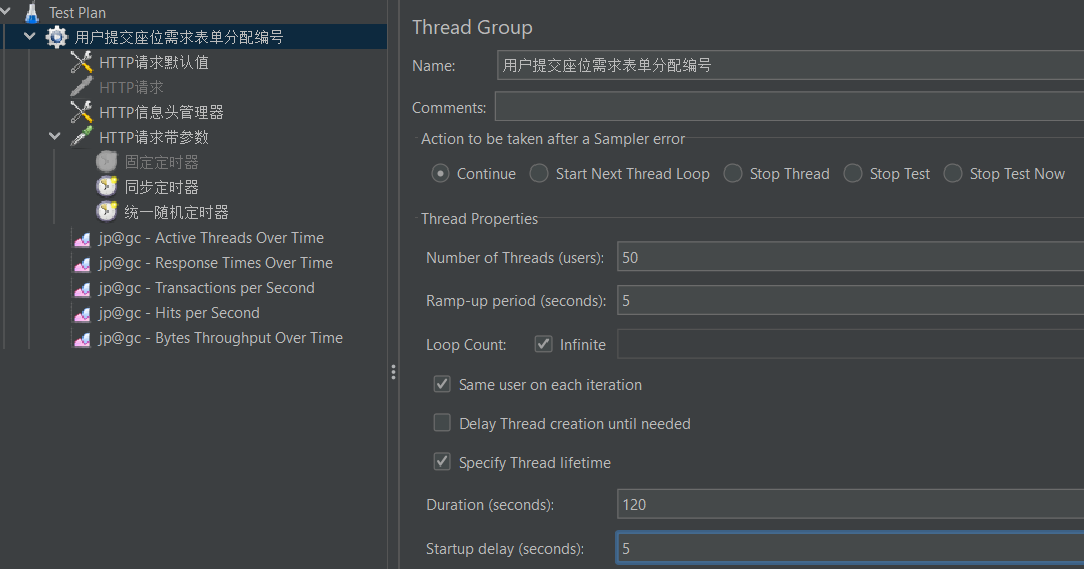
所有用例都能得到正确的预期结果，说明程序设计较为完善。

## 性能测试

测试工具：JMeter测试

* + 1. 用户提交座位需求表单分配

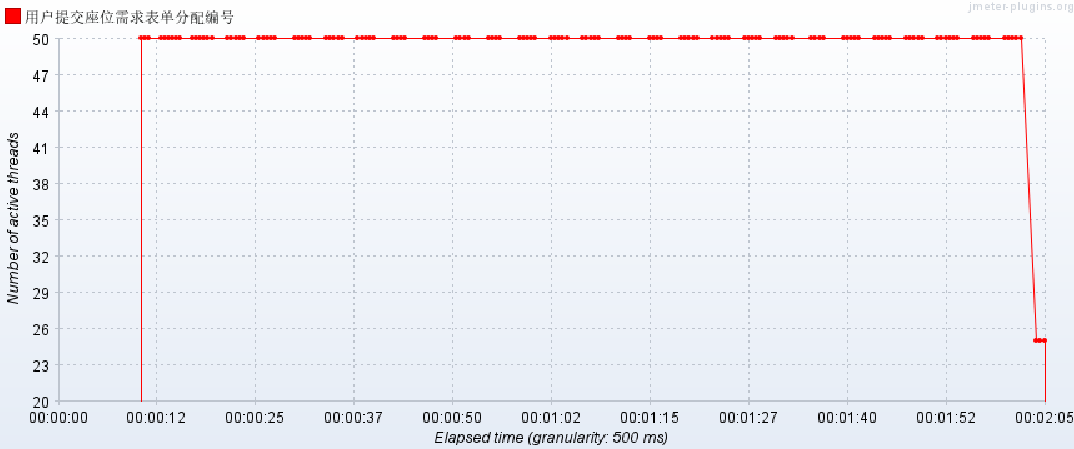
1. 场景测试设计



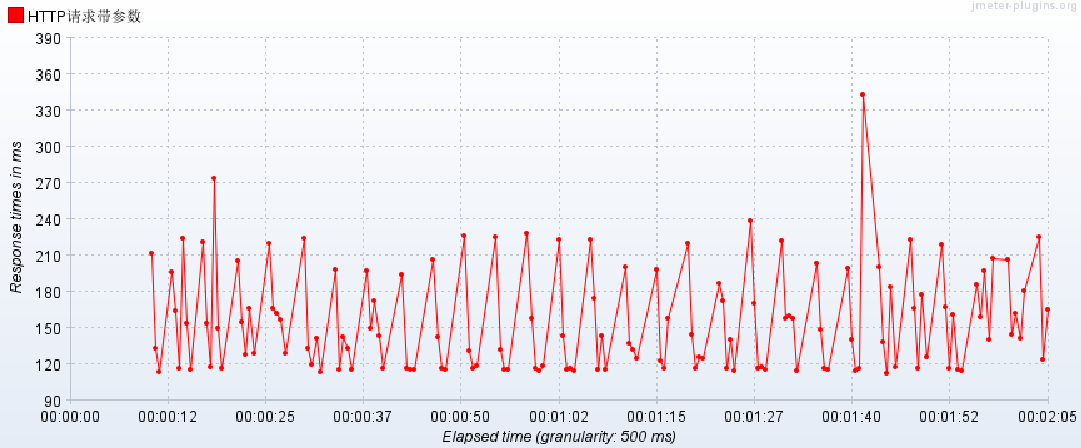
此场景用于测试用户通过POST提交座位需求表单时后端分配编号的过程。

(2) 场景记录

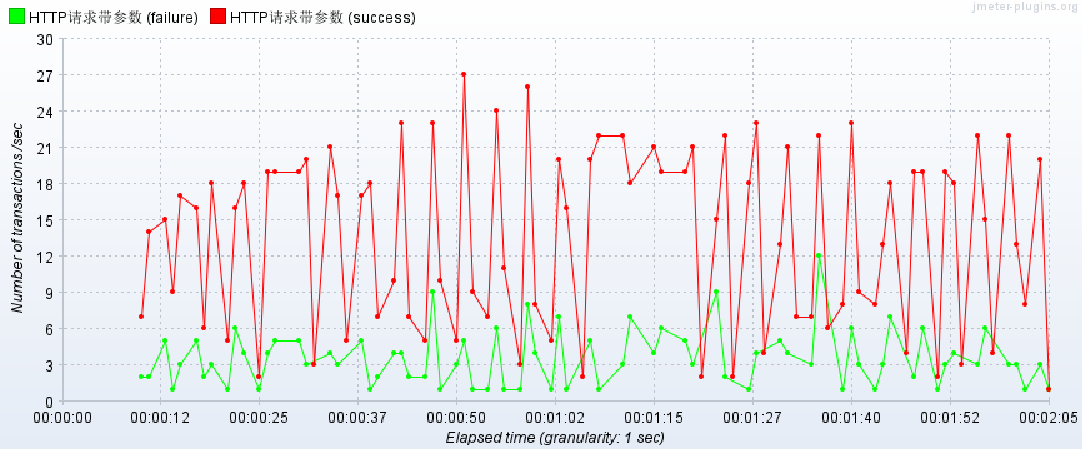
Active Threads Over Time：



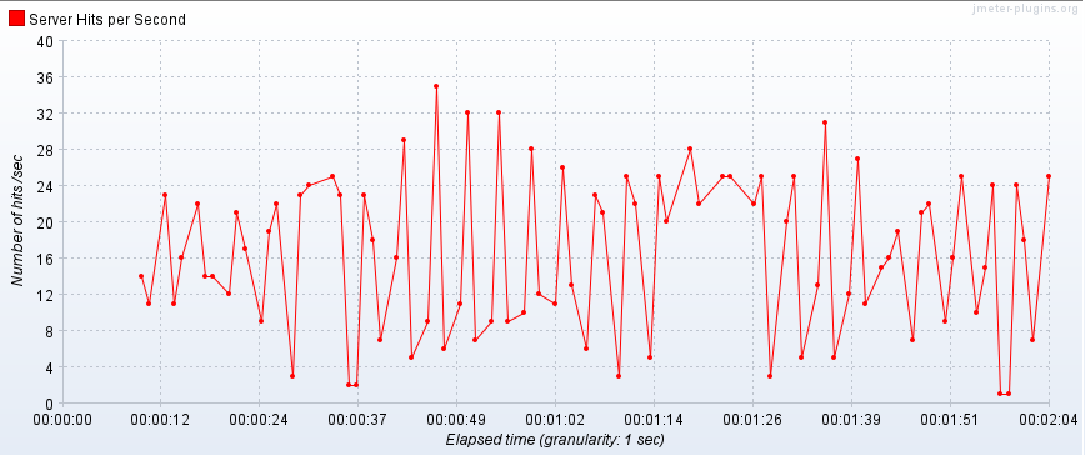
Response Times Over Time：



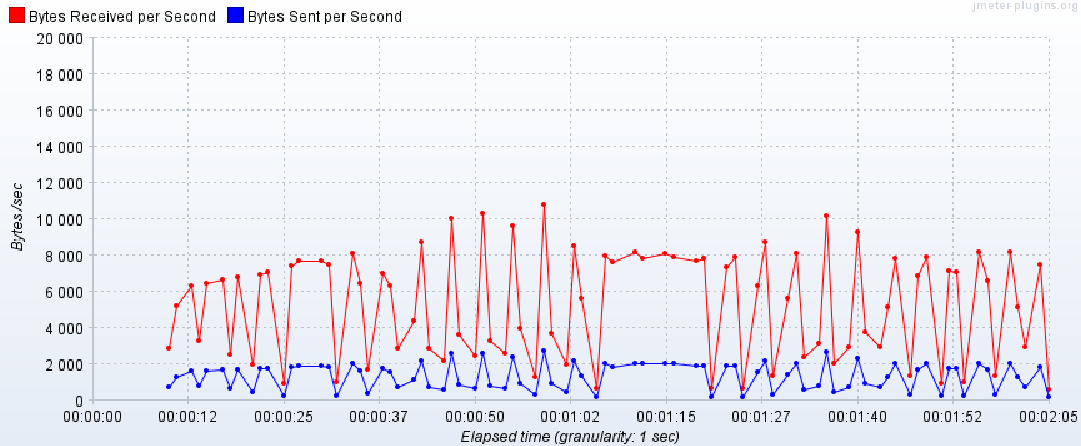
Transactions per Second：



Hit per Second：



Bytes Throughout Over Time



1. 分析结果：



# 缺陷分析

## 缺陷报告

|  |  |  |  |
| --- | --- | --- | --- |
| 缺陷ID | BUG001 | | |
| 测试软件名称 | 平凡餐馆信息管理系统 | | |
| 缺陷描述 | 使用Cookie保存信息可能会导致数据库被清空后产生意外的请求信息 | | |
| 测试版本 | 1.0 | 附件 | 无 |
| 缺陷发现日期 | 2024.6.7 | 缺陷严重程度 | 严重 |
| 测试人员 | 吴平凡 | 缺陷优先级 | 立即解决 |
| 测试环境 | 处理器：Intel | IDE:IDEA |  |
| 重现步骤 | 1. 进入排队正常选座页面 2. 选座成功 3. 此时成功保存了cookie 4. 数据库信息被清空 5. 再次访问原始页面会进入点菜界面 | | |
| 备注 | 在后端检测到异常后给前端发送重定向回复 | | |

# 测试总结

在对平凡餐馆信息管理系统的测试过程中，我们进行了全面的测试，涵盖了白盒测试、黑盒测试以及性能测试。以下是各个测试阶段的总结：

## **白盒测试**：

**基本路径覆盖**：验证了点菜服务层DishService接口的dishOrder方法能否正确处理订单的创建和保存。测试结果显示，该方法在各种输入条件下均表现正确。

**条件覆盖**：验证了顾客服务层GuestServer接口的guestCome方法能否正确处理顾客的到来。测试结果显示，该方法在各种输入条件下均表现正确。

**语句覆盖**：验证了顾客服务层GuestServer接口的selectTable方法能否正确处理顾客选择桌子的操作。测试结果显示，该方法在各种输入条件下均表现正确。

## **黑盒测试**：

**获取所有桌子信息**：测试了系统能否正确返回所有桌子的信息，并设置有效的Cookie。测试结果显示，所有用例均能得到预期结果。

**选择桌子**：测试了用户选择桌子后，系统能否更新桌子的状态，并重定向到菜单页面。测试结果显示，所有用例均能得到预期结果。

**提交排队表单**：测试了用户提交排队表单后，系统能否正确分配顾客编号并设置相应的Cookie。测试结果显示，所有用例均能得到预期结果。

## **性能测试**：

使用JMeter进行测试，模拟了大量用户同时提交座位需求表单的场景。测试结果显示，系统在高负载下表现稳定，响应时间和吞吐量均符合预期。

## 结论

通过此次测试，平凡餐馆信息管理系统的各项功能在各种测试场景下均表现良好，满足需求说明书中的各项要求。系统在功能正确性、稳定性和用户体验方面均达到了预期目标，具备上线条件。