<Mardrobe智能穿衣搭配系统>

测试报告

版本 <3.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <12/7/2019> | <1.0> | <第一次迭代测试> | <张帅 李佳睿> |
| <19/7/2019> | <3.0> | <第三次迭代测试> | <张帅 李佳睿> |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 测试概要 4

3. 测试环境 4

4. 测试结果及分析 4

4.1 需求覆盖率及缺陷分布 4

4.2 缺陷严重程度 5

5. 缺陷清单 5

5.1 功能性缺陷 5

5.2 非功能性缺陷 6

6. 测试结论与建议 6

测试报告

# 简介

## 目的

测试第三次迭代产品的前后端功能和数据交互。

## 范围

相关项目：Mardrobe第三次迭代产品。

影响事物：后续的工作安排以及产品功能调整，迭代完成后进行下一次与核心功能对接的迭代。

## 定义、首字母缩写词和缩略语

Mardrobe ——Mardrobe智能穿衣搭配系统

## 参考资料

React native中文文档等

## 概述

针对第三次迭代的简介、测试概要、测试环境与测试结果分析。

# 测试概要

时间：7.15-7.19

地点：软件学院3108

人员：张帅 李佳睿

测试方法：jest框架测试 junit框架测试 手动测试

测试内容：功能测试 界面测试 单元测试

# 测试环境

软件：前端在Android Studio中使用jest测试react native；后端在Intellij idea中使用junit测试spring cloud。

硬件：后端功能测试在PC端进行，前端页面测试通过移动端真机进行。

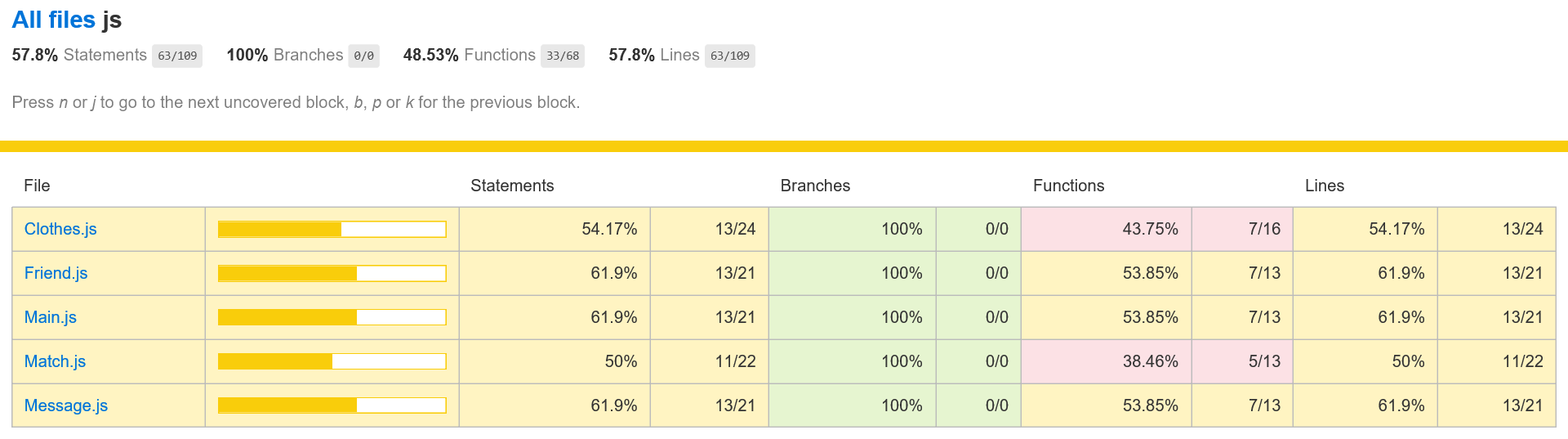
# 测试结果及分析

测试功能点数：8

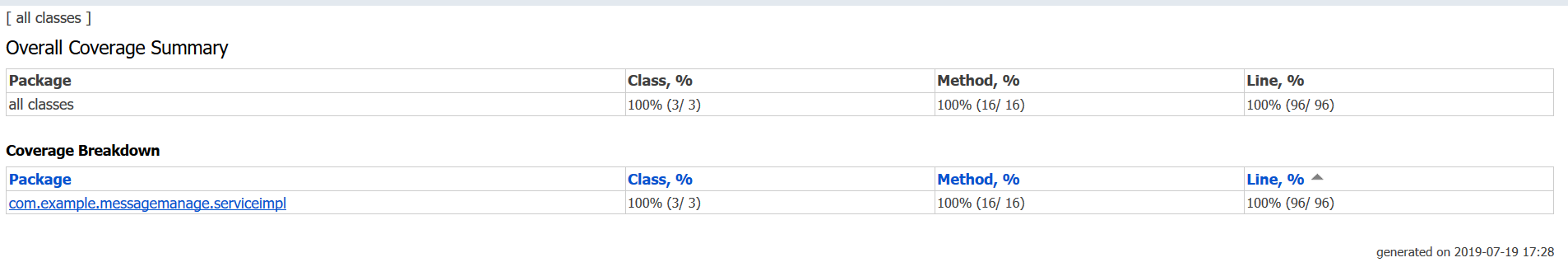
测试用例数：27

缺陷数：0

前端：



后端：



## 需求覆盖率及缺陷分布

**表1 需求覆盖率及缺陷分布**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试内容** | | **测试**  **用例数** | **需求**  **覆盖率** | **缺陷数** | **缺陷率** | **备注** |
| **功能项** | **查看社区动态** | 4 | 8.3% | 0 | 0 | 包括前端和后端，数据库mongodb |
| **管理动态** | 4 | 16.7% | 0 | 0 | 包括前端和后端，数据库mongodb |
| **查看好友** | 1 | 8.3% | 0 | 0 | 包括前端和后端，数据库mongodb |
| **好友管理** | 4 | 16.7% | 0 | 0 | 申请/添加/删除 |
| **查看私信** | 3 | 8.3% | 0 | 0 | 包括前端和后端，数据库mongodb |
| **管理私信** | 4 | 16.7% | 0 | 0 | 通过好友/私信页面 |
| **查看衣柜** | 3 | 8.3% | 0 | 0 | 包括前端和后端，数据库mongodb |
| **管理衣柜** | 4 | 16.7% | 0 | 0 | 添加/删除 |
| **功能项小计** | 27 | 100% | 0 | 0 | 版本二的功能已全部完成，搭配推荐功能独立进行，将在后续版本相结合 |
| **非功**  **能项** | **易用性** | 1 | 100% | 0 | 0 | 未经培训过的用户可在5分钟内熟悉各页面、组件的操作 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | **非功能项小计** | 1 | 100% | 0 | 0 | 当前版本功能较为简单，故只做易用性测试，其他非功能测试在后续版本继续完善 |
|  | **总计** | 13 | 100% | 0 | 0 | 完成第二次迭代计划 |

## 缺陷严重程度

**表2 缺陷严重程度**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **严重程度** | **致命** | **严重** | **一般** | **细微** | **总数** |
| **缺陷个数** | 0 | 0 | 0 | 0 | 0 |
| **占缺陷百分比** | 0 | 0 | 0 | 0 | 0 |

# 缺陷清单

## 功能性缺陷

**表3 功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **功能模块** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

表中相关项说明：

1. 缺陷编号：为每个缺陷赋予一个唯一的编号，可以通过此编号对缺陷进行跟踪。例如：Bug001。
2. 缺陷严重程度：缺陷可以根据严重程度分为以下几种情况。
3. 致命（fatal）：致命的错误，测试执行直接导致系统死机、蓝屏、挂起、或是程序非法退出；系统的主要功能或需求没有实现。
4. 严重（critical）：严重错误，系统的次要功能点或需求点没有实现；数据丢失或损坏。执行软件主要功能的测试用例导致系统出错，程序无法正常继续执行；程序执行过于缓慢或是占用过大的系统资源。
5. 一般（major）：不太严重的错误，这样的缺陷虽然不影响系统的基本使用，但没有很好地实现功能，没有达到预期的效果。如次要功能丧失，界面错误，打印内容、格式错误，提示信息不太正确，或用户界面太差，简单的输入限制未放在前台进行控制，删除操作未给出提示，操作时间长等。
6. 细微（minor）：一些小问题，对功能几乎没有影响，产品及属性仍可使用。如软件的实际执行过程与需求有较小的差异；程序的提示信息描述容易使用户产生混淆；有个别错别字、 文字排列不整齐；辅助说明描述不清楚，显示格式不规范，长时间操作未给用户进度提示，提示窗口文字未采用行业术语，可输入区域和只读区域没有明显的区分标志，系统处理未优化等。
7. 功能模块：所测试并出现该缺陷的功能模块名称。
8. 缺陷标题：描述缺陷的标题。
9. 缺陷描述：对缺陷的详细描述。描述缺陷的重现步骤和问题所在，描述需清晰明了，使研发人员可以按照描述将缺陷重现出来。必要时可以放上相应的截图，以便于研发人员理解问题所在。

## 非功能性缺陷

**表4 非功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **测试类型** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 |  |  | 性能 |  |  |  |
| 2 |  |  | 性能 |  |  |  |
| … |  |  | … |  |  |  |
| n |  |  | 可靠性 |  |  |  |

# 测试结论与建议

**问题：**

在前端测试中，覆盖率不高。

原因分析：部分页面的操作灵活性较强，难以体现到具体的测试用例中，并且其操作和界面一致性关联较大，对视图要求较高，又无需大量重复，故覆盖率测试只测试渲染页面的正确性，其他操作采用手动测试。除此之外，前端的功能均需要通过后端的函数实现，因此将主要的单元测试放在后端。

**结论：**

在此次迭代测试中，所测界面和功能预期结果，但性能有待提高。

**建议措施：**

1.后续工作中，进行和搭配推荐算法的对接，实现完整的软件架构。

2.在保证功能完整正确的基础上，对性能进行调优。