<有课>

测试报告

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <日/月/年> | <x.x> | <详细信息> | <姓名> |
| 2019.7.8 | 0.2 | 迭代一测试报告 | 郭志 |
|  |  |  |  |
|  |  |  |  |

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

*本文档是有课项目开发的第一次迭代的测试报告，是对从2019/7/1-2019/7/10之间，对于有课项目的测试形成的工作报告。*

## 范围

《测试用例》 C18有课项目

## 定义、首字母缩写词和缩略语

无

## 参考资料

《立项建议书》 C18有课项目

《用例规约》 C18有课项目

《迭代计划》(1) C18有课项目

《测试用例》 C18有课项目

## 概述

*测试报告包括测试的目的，范围，参考资料，测试概要，测试环境，测试的结果与分析，需求覆盖率及缺陷分布，缺陷清单，测试结论。其中测试结果与分析将从前后端两个方面来分别阐述。*

# 测试概要

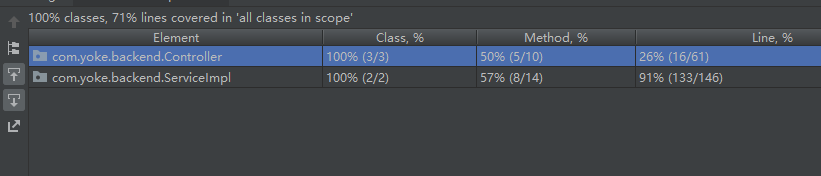
测试贯穿与开发全过程，以测试驱动开发，测试参与人员即是测试开发人员。测试方法主要为单元测试，测试内容包括前端react-native测试与后端spring测试。

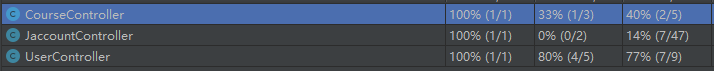
# 测试环境

第一次迭代尚未部署到服务器，仅在开发人员的本机上测试。

# 测试结果及分析

对于后端主要代码的测试：使用Junit4





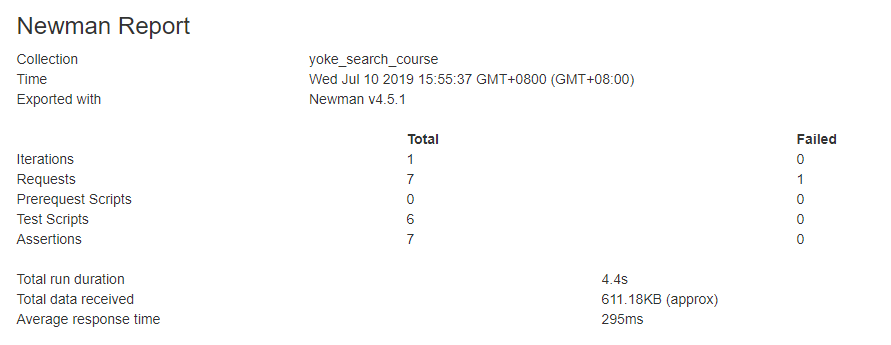


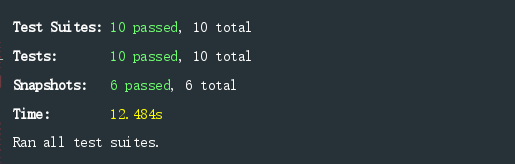
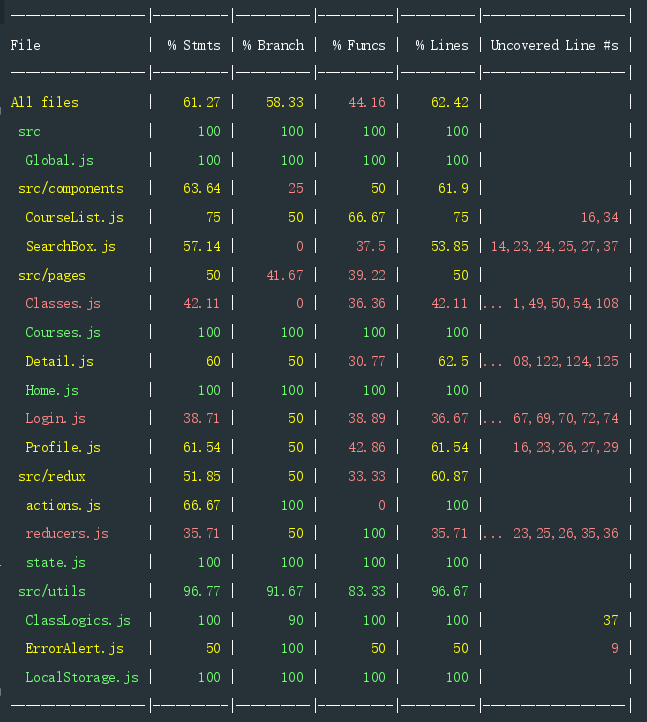
对于controller的测试中由于我们有一个JaccountController需要一个从网络信息中心传回的code作为参数，不适合测试，因此测试代码覆盖率较低。

对于serviceImpl，对主要的CourseServiceImpl,UserServiceImpl都做了详细测试，代码覆盖率达到了91%。

对于后端接口的测试：使用Postman的newman测试接口

课程查询接口：



对于前端代码的测试： 使用Jest 框架测试

对所有渲染页面使用快照Snapshot测试；将业务逻辑从组件中抽离并模块化，对逻辑使用逻辑测试。

## 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** | 课程查询 | 6 | 100% | 1 | 13% | 当查询参数为空时会返回所有的课程信息，这对项目性能有影响，需要更改 |
| 用户获取课程详细信息 | 2 | 100% | 0 | 0 |  |
| 用户登录 | 1 | 70% | 0 | 0 | 由于用户Jaccount登录需要调用特殊接口，获得特定参数，没有进行测试 |
| 导入课程数据 | 1 | 100% | 5 | 20% | 缺陷主要产生于解析课程数据时，例如将课程学分转为了整型，缺少了课程备注。 |
|  | 查看课程表 | 1 | 100% | 0 | 0% |  |
| **非功**  **能项** | **性能** |  |  |  |  |  |
| **可靠性** |  |  |  |  |  |
| **…** |  |  |  |  |  |
|  | **非功能项小计** |  |  |  |  |  |
|  | **总计** | *11* | *97%* | *7* | *9%* |  |

## 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 | 4 | 1 | 5 |
| **占缺陷百分比** | 0% | 0% | 80% | 20% | 100% |

# 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 | Bug1 | 一般 | 课程查询 | 课程详情页退出缓慢 | 打开课程详情页选择退出时，需要1s才能退出到查询页面 | Testcase-003 |
| 2 | Bug2 | 一般 | 课程查询 | 课程详情有异常信息 | 课程单双周信息显示出？？？ | Testcase-003 |
| 3 | Bug3 | 一般 | 课程信息导入 | 课程导入步骤不够自动化 | 由于教务处不提供课程接口，需要拿到最新的url和cookies传给程序，让其得到课程信息 | Testcase001 |
| 4 | Bug4 | 轻微 | 课程信息导入 | 课程信息导入时的完整性 | 课程信息导入时备注未考虑进去 | Testcase001 |
| 5 | Bug5 | 一般 | 用户登录 | 登陆完成，没有过渡页面 | Jaccount登陆完成，并返回App后，App会在login界面停留短暂时间，此时使用过渡页更为合理 | Test-007 |

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。

# 测试结论与建议

本次测试主要是针对于迭代一要实现功能的功能测试，尚未实现性能测试，可靠性测试。通过不断地测试与开发，修改了很多潜在的bug，对于项目的主逻辑部分测试覆盖率较高。这次测试主要针对的模块是迭代一中课程导入，用户登录，查看课表，搜索课程，查看课程这些较为基础的功能。之后的功能需要在此基础上拓展，测试为之后的开发提供了保障。下一次迭代我们应该更强调以测试驱动开发，使用自动化测试工具，让前后端的测试更轻松。