即应匹配信息发布平台

测试报告

版本 <1.0>

[注：其中包括用方括号括起来并以蓝色斜体（样式=InfoBlue）显示的文本，它们用于向作者提供指导，在发布此文档之前应该将其删除。按此样式输入的段落将被自动设置为普通样式（样式=Body Text）。]

[要定制 Microsoft Word 中的自动字段（选中时显示灰色背景），请选择 File >Properties，然后将 Title、Subject 和 Company 等字段替换为此文档的相应信息。关闭该对话框后，通过选择 Edit> Select All（或 Ctrl-A）并按 F9，或只是在字段上单击并按 F9，可以在整个文档中更新自动字段。对于页眉和页脚，这一操作必须单独进行。按 Alt-F9，将在显示字段名称和字段内容之间切换。有关字段处理的详细信息，请参见 Word 帮助。]

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> |
| <09/07/2019> | <1.0> | 迭代一测试报告 | Summer855小组 |
| <10/07/2019> | <1.1> | 迭代一测试报告 | Summer855小组 |
|  |  |  |  |

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

本测试报告是 用来记录“即应匹配信息发布平台”开发过程中的功能及非功能需求的完成情况，主要目的是发现设计和实现中的缺陷和不足，然后进行修复和改善。

## 范围

此测试报告撰写于迭代一功能完成之后，测试范围包含迭代一的功能测试，包括登录，用户信息更改，注册。此测试报告基于测试完成之后，对测试中使用的方法、测试的有效程度进行总结

## 定义、首字母缩写词和缩略语

即应——即应匹配信息发布平台

Auth service —— 即应微服务架构中的用于登录、验证的服务

User service —— 即应微服务架构中用于用户信息的添加、修改、查询、删除的服务

Api gateway —— 即应微服务架构中用于禁停http请求，将请求转发，调用其他服务的服务

## 参考资料

《即应项目软件需求规约》 v1.2

《即应第一次迭代计划》 v1.0

《即应匹配信息发布平台 第一次迭代 系统测试用例》 v1.0

## 概述

本测试报告包含测试概要、测试环境、测试结果及分析、测试缺陷报告等，由于第一次迭代尚未进行非功能性需求的分析设计，此次测试将不会对性能等非功能性需求进行测试，重点将会放在登录、用户信息管理、注册等功能上

# 测试概要

测试时间：09/07/2019

地点：项目开发教师（软件学院3101教师）

人员：Auth service, user service开发者、微信小程序开发者、api gateway开发者

测试方法：单元测试，人工测试

测试内容：对涉及验证等逻辑功能进行单元测试，确保这些逻辑符合我们的预期，对于项目迭代一所要求的功能进行人工测试，用postman发送请求、微信小程序点击等方法测试系统反馈的内容是否符合预期

# 测试环境

系统后台发布在ubuntu16.04上，CPU4核，内存8G，用kubernetes进行容器管理。Kubernetes版本是v1.15.0

微信小程序尚在开发模式，使用微信开发者工具与后台的交互，微信开发者工具运行在windows10家庭版上，版本是稳定版 Stable Build (1.02.1904090)

Postman是运行在linux Ubuntu 18.04 TLS上，版本是v7.2.2

# 测试结果及分析

测试功能点数：5， 分别是 第一次登录小程序，扫描二维码绑定微信ID和jaccount， 扫描二维码后返回小程序成功登录，退出登录，修改用户信息

测试用例数：15 （不包括单元测试）

缺陷数：5

测试用例：详见[《即应匹配信息发布平台 第一次迭代 系统测试用例》](测试用例.xls)

## 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** |
| **1** | **第一次登录小程序** | 3 | 100% | 2 | 66.7% |  |
| **绑定jaccount和微信id** | 1 | 100% | 0 | 0 |  |
| **扫描二维码然后返回小程序** | 1 | 100% | 0 | 0 |  |
| **退出登录** | 1 | 100% | 0 | 0 |  |
| **用户信息修改** | 2 | 100% | 1 | 50% |  |
| **功能项小计** | 5 | 100% | 0 | 0 |  |
| **非功**  **能项** | **性能** |  |  |  |  |  |
| **可靠性** |  |  |  |  |  |
| **…** |  |  |  |  |  |
|  | **非功能项小计** |  |  |  |  |  |
|  | **总计** |  |  |  |  |  |

## 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** | 2 | 1 | 0 | 0 | 0 |
| **占缺陷百分比** | 66.7% | 33.3% | 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 | Bug001 | Urgent(严重) | 第一次登录 | 后台返回401，未返回二维码 | 后台未对err进行判断，导致内存溢出 | jing-testcase-017 |
| 2 | Bug002 | Urgent(严重) | 第一次登录 | 后台返回400，未返回二维码 | 后台API gateway中对于查询参数的获取出现问题，应该调用Query，但是使用了Params | jing-testcase-018 |
| 3 | Bug003 | High(主要) | 用户信息更改 | 后台传入参数出现混乱 | 后台传入参数出现混乱，导致用户信息中用户名和签名存入数据库时颠倒了 | jing-testcase-023 |
| 4 |  |  |  |  |  |  |

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 |  |  |  |

本次未做非功能性的测试，因此未发现非功能性缺陷

# 测试结论与建议

微服务架构对于各个微服务进行单元测试比较容易，对每个服务里的对应的方法进行单元测试即可。

但是集成测试的时候就比较麻烦，目前我们使用的是发布之后使用前台与后台进行交互，即使用人工点击、填写信息等。这样的方式覆盖率并不是很完整，而且获取反馈也比较慢，更改代码之后提交还需要等待CI进行单元测试之后。

下一步中，我们将进行更加完整的的单元测试和系统的集成测试，争取能够改进集成测试的方法，以提高工作效率，法相更多的问题。