目录

[软件测试报告(STR) 1](#_Toc235939334)

[1引言 3](#_Toc235939335)

[1.1标识 3](#_Toc235939336)

[1.2系统概述 3](#_Toc235939337)

[1.3文档概述 3](#_Toc235939338)

[2引用文件 3](#_Toc235939339)

[3测试结果概述 3](#_Toc235939340)

[3.1对被测试软件的总体评估 3](#_Toc235939341)

[3.2测试环境的影响 4](#_Toc235939342)

[4详细的测试结果 4](#_Toc235939344)

[5测试记录 4](#_Toc235939345)

[6评价 5](#_Toc235939346)

[7测试活动总结 5](#_Toc235939347)

# 1引言

## 1.1标识

本文档测试的内容是imSoNervousAh团队所作的anervouswebsite项目,一个用于实现清华大学校内微信公众号自助管理的网站。

## 1.2系统概述

本项目为微信公众号管理平台，甲方为清华大学。在软件工程课程中隶属于PRJ1。

本项目可以为学校提供管理校园内学生组织公众号的网页端平台，学生用户可以用校园网账号登录本平台进行公众号的备案申请，管理员用户则可以审批申请、查看各公众号的信息并进行数据统计与分析。此外，管理员和学生用户还可以通过“站内信”进行交流沟通。

## 1.3文档概述

本文档主要介绍了本小组关于anervouswebsize项目的测试步骤、流程。

# 2引用文件

主要参考了给的样例中的软件测试报告(STR)。

# 3测试结果概述

本章应分为以下几条提供测试结果的概述。

## 3.1对被测试软件的总体评估

在本项目中，测试的代码主要集中于/database/tests.py内，即对后端的database的测试。

主要有

StudentTestCase,AdminTestCase,ApplicationTestCase,ArticleTestCase,MessageTest,NervousTest这么几个单元。

经过测试，结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 被测用例 | 用例总数 | 通过用例数 | 通过用例比例 |
| StudentTestCase | 5 | 5 | 100% |
| AdminTestCase | 5 | 5 | 100% |
| ApplicationTestCase | 9 | 9 | 100% |
| ArticleTest | 3 | 3 | 100% |
| MessageTest | 3 | 3 | 100% |
| NervousTest | 7 | 7 | 100% |

## 3.2测试环境的影响

在不同的django版本下，覆盖率存在差异，在django1.8.7下覆盖率达到了68%。对于个别的django版本，可以达到80%以上。

# 4详细的测试结果

本章应分为以下几条提供每个测试的详细结果。

**4.1 StudentTestCase**

在StudentTestCase中，主要完成了对backend.check\_student\_information\_filled接口函数的测试，这个函数主要用来检查学生的身份信息是否填写完备。

我们划分其等价类为空的情况，未填写完全的情况，以及填写完全的情况，并分别进行了测试。最后，我们也测试了backend.get\_student\_by\_id接口，这个接口从数据库中按照id为秩获取学生用户信息并返回。

**4.2 AdminTestCase**

在AdminTestCase中，我们主要完成了对管理员账户的相关测试。

整个测试流程包括调用backend.add\_admin往数据库内加入一个新的管理员账号，加入一个相同id的账号以检测数据库是否会甄别相同id的管理员。调用backend.del\_admin来往数据库中删除一个刚刚添加进的管理员。再次删除以检验数据库是否已经完全删除这个id，并且数据库是否会检验删除一个不存在的账号。为了模拟和测试管理员的登陆，我们添加了一个新用户并分别用正确和错误的密码来测试数据库的反馈。

**4.3 ApplicationTestCase**

在ApplicationTestCase中，我们主要完成了对公众号的相关部分的测试。

在测试的setUp阶段，调用backend.add\_application接口进行测试，添加一个全新公众号。

再次调用backend.add\_application接口，以检测数据库是否可以检测包含相同wx\_id的公众号。

同时，一个申请，可以用各种键值来取得。我们分别测试了使用提交用户(user\_submit)，提交状态(status，包括pending,rejected等等)，审批管理员等来测试一个申请是否可以被数据库检索到。

申请的状态可以被改变，在测试中，我们分为学生修改了状态和管理员修改了状态来分别测试。调用backend.recall\_application来模拟学生撤回申请，并修改内容后，调用backend.student\_modify\_application来完成这次修改。为了模拟管理员的审批流程，调用backend.modify\_application进行模拟修改，将申请的状态改为approved，并再次获取approvd的申请，以检验是否已经完全通过。

调用backend.recall\_application，再调用backend.del\_application来模拟一个申请的删除。最后在数据库中再次检验这个申请以确认是否完全删除。

**4.4 ArticleTest**

在ArticleTest中，我们主要测试了收录的文章的相关接口。包括获取全部的收录的文章与通过公众号的id来检索文章。

二者是分别通过调用backend.get\_aritcles和backend.get\_articles\_by\_official\_account\_id来完成的。

**4.5 MessageTest**

在MessageTest中，我们主要测试了站内信的相关接口的工作是否正常。

调用backend.get\_messages获取所有的站内信，以检验此接口是否正常工作。

调用backend.add\_message添加一个管理员与学生之间的站内信。首先先调用backend.add\_admin添加了一个新的管理员。

**4.6 NervousTest**

在NervousTest中，我们主要测试了一些相对来说比较杂的单元。

测试调用了backend.get\_lastest\_record\_date来检测关于公众号的信息是否正确。这个测试在一般情况下是检验函数是否能正常运行而不出错。

调用了backend.del\_official\_account来检测删除一个关注的公众号的结果。再次调用这个接口来检测是否已经正确删除。

调用了

backend.check\_forewarn\_rule\_on\_account,backend.check\_forewarn\_rule,backend.forewarn\_rule\_from\_dict，backend.check\_all\_foreign\_rules这四个函数来检验预警的相关接口函数是否正确。在这里主要测试的是这些预警规则在数据库里的存储是否正确，关于触发预警规则之后的预警邮件的测试我们进行了单独的测试——主要是考虑到频繁的测试可能会导致过多的垃圾邮件。

**4.7 性能测试**

同时，我们还进行了性能测试。在backend/utils.py底下，有一个build\_large\_test\_db()函数。利用这个函数进行短时间内500次的申请，10000次的文章读取操作，用时为26086.661ms。测试系统是OS X EI Capitan 10.11.2，MacBook Pro(Retina, 15-inch, Mid 2015)。

可以看出，这样子的大规模的操作的情况下，整个网站性能只能算是一般。这个数据是在本地测试的，在实际部署到服务器上时，效率应该会有所提高。

# 5测试记录

|  |  |  |
| --- | --- | --- |
| 测试时间 | 测试内容 | 测试人员 |
| Sprint2 | 简单的前后端对接测试 | 韦毅龙，胡泽聪，杨天龙，黄豪硕 |
| Sprint3 | 初步的单元测试，代码阅读，测试后端函数 | 韦毅龙，黄豪硕 |
| Sprint4 | 完善单元测试，提高覆盖率，代码阅读，进行功能测试。 | 黄豪硕 |

# 6评价

本小组对整个项目进行了测试。从测试结果来看，整个项目还是比较顺利的。

在我们的测试过程中，有在代码层面上的测试，如代码覆盖率上的测试，通过对不同的单元的测试，我们测试了基本功能的正确性。

也有实际功能上的人工测试，比如我们自己在网页上手动模拟了许多不合法的请求，收录了很多的公众号来检测。

总而言之，整个项目的测试还是比较顺利的。

# 7测试活动总结

本次项目主要是消耗了一些人力和时间进行测试，测试的过程本身并没有特别厉害的技术，主要是一些细节上的测试需要思考清楚。